

Image Parsing: Unifying Segmentation, Detection, and Recognition

Zhuowen Tu¹, Xiangrong Chen¹, Alan L. Yuille^{1,2}, and Song-Chun Zhu^{1,3}

Departments of Statistics¹, Psychology², and Computer Science³,
University of California, Los Angeles, Los Angeles, CA 90095.
emails: {ztu,xrchen,yuille,sczhu}@stat.ucla.edu

Abstract

In this paper we present a Bayesian framework for parsing images into their constituent visual patterns. The parsing algorithm optimizes the posterior probability and outputs a scene representation in a “parsing graph”, in a spirit similar to parsing sentences in speech and natural language. The algorithm *constructs* the parsing graph and *re-configures* it dynamically using a set of reversible Markov chain jumps. This computational framework integrates two popular inference approaches – *generative* (top-down) methods and *discriminative* (bottom-up) methods. The former formulates the posterior probability in terms of generative models for images defined by likelihood functions and priors. The latter computes discriminative probabilities based on a sequence (cascade) of bottom-up tests/filters. In our Markov chain design, the posterior probability, defined by the generative models, is the invariant (target) probability for the Markov chain, and the discriminative probabilities are used to construct proposal probabilities to drive the Markov chain. Intuitively, the bottom-up discriminative probabilities activate top-down generative models. In this paper, we focus on two types of visual patterns – generic visual patterns, such as texture and shading, and object patterns including human faces and text. These types of patterns compete and cooperate to explain the image and so image parsing unifies image segmentation, object detection, and recognition (if we use generic visual patterns only then image parsing will correspond to image segmentation [46]). We illustrate our algorithm on natural images of complex city scenes and show examples where image segmentation can be improved by allowing object specific knowledge to disambiguate low-level segmentation cues, and conversely object detection can be improved by using generic visual patterns to explain away shadows and occlusions.

Keywords. Image parsing, image segmentation, object detection, object recognition, data driven Markov Chain Monte Carlo, AdaBoost.

1 Introduction

1.1 Objectives of Image Parsing

We define image parsing to be the task of decomposing an image \mathbf{I} into its constituent visual patterns. The output is represented by a hierarchical graph W — called the “parsing graph”. The goal is to optimize the Bayesian posterior probability $p(W|\mathbf{I})$. Figure 1 illustrates a typical example where a football scene is first divided into three parts at a coarse level: a person in the foreground, a sports field, and the spectators. These three parts are further decomposed into nine visual patterns in the second level: a face, three texture regions, some text, a point process (the band on the field), a curve process (the markings on the field), a color region, and a region for nearby people. In principle, we can continue decomposing these parts until we reach a resolution criterion. The parsing graph is similar in spirit to the parsing trees used in speech and natural language processing [32] except that it can include horizontal connections (see the dashed curves in Figure 1) for specifying spatial relationships and boundary sharing between different visual patterns.

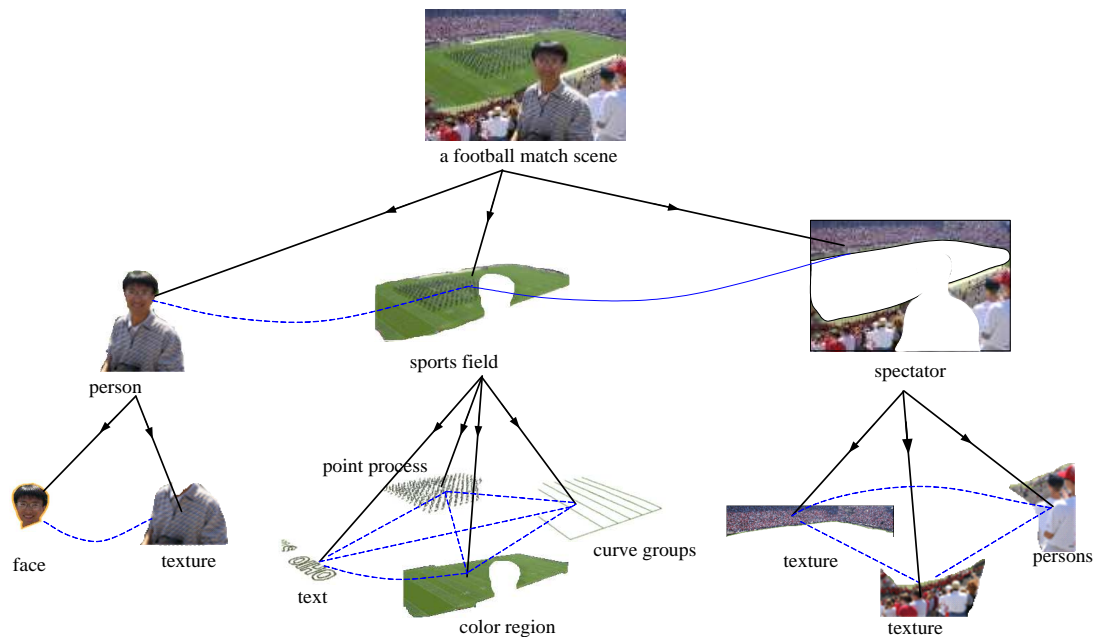


Figure 1: Image parsing example. The parsing graph is hierarchical and combines generative models (downward arrows) with horizontal connections (dashed lines), which specify spatial relationships between the visual patterns. See Figure 4 for a more abstract representation including variables for the node attributes.

As in natural language processing, the parsing graph is not fixed and depends on the input

image(s). An image parsing algorithm must *construct* the parsing graph on the fly¹. Our image parsing algorithm consists of a set of reversible Markov chain jumps [21] with each type of jump corresponding to an operator for *reconfiguring* the parsing graph (i.e. creating or deleting nodes or changing the values of node attributes). These jumps combine to form an ergodic and reversible Markov chain in the space of possible parsing graphs. The Markov chain probability is guaranteed to converge to the invariant probability $p(W|\mathbf{I})$ and the Markov chain will simulate fair samples from this probability.² Our approach is built on previous work on Data-Driven Markov Chain Monte Carlo (DDMCMC) for recognition [57], segmentation [46], grouping [47] and graph partitioning [1, 2].

Image parsing seeks a full generative explanation of the input image in terms of generative models, $p(\mathbf{I}|W)$ and $p(W)$, for the diverse visual patterns which occur in natural images, see Figure 1. This differs from other computer vision tasks, such as segmentation, grouping, and recognition. These are usually performed by isolated vision modules which only seek to explain parts of the image. The image parsing approach enables these different modules to cooperate and compete to give a consistent interpretation of the entire image.

The integration of visual modules is of increasing importance as progress on the individual modules starts approaching performance ceilings. In particular, work on segmentation [43, 46, 17] and edge detection [26, 8] has reached performance levels where there seems little room for improvement when only low-level cues are used. For example, the segmentation failures in Figure 2 can only be resolved by combining segmentation with object detection and recognition. There has also recently been very successful work on the detection and recognition of objects [29, 53, 41, 4, 52, 54] and the classification of natural scenes [3, 38] using, broadly speaking, discriminative methods based on local bottom-up tests.

But combining different visual modules requires a common framework which ensures consistency. Despite the effectiveness of discriminative methods for computing scene components, such as object labels and categories, they can also generate redundant and conflicting results. Mathematicians have argued [6] that discriminative methods must be followed by more sophisticated processes to

¹Unlike most graphical inference algorithms in the literature which assume fixed graphs, see belief propagation [55].

²For many natural images the posterior probabilities $P(W|\mathbf{I})$ are strongly peaked and so stochastic samples are close to the maxima of the posterior. So in this paper we do not distinguish between sampling and inference (optimization).

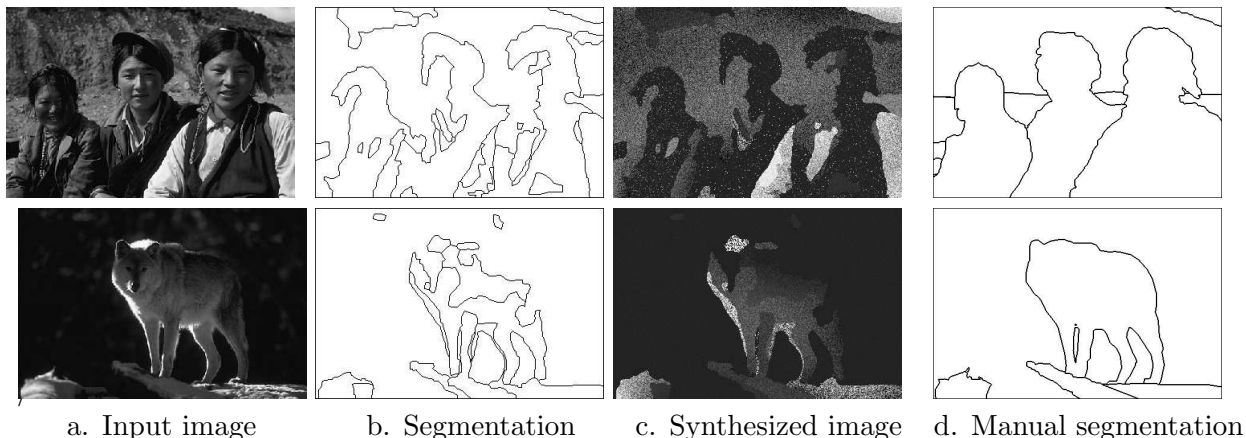


Figure 2: Examples of image segmentation failure by an algorithm [46] which uses only generic visual patterns (i.e. only low-level visual cues). The results (b) show that low-level visual cues are not sufficient to obtain good intuitive segmentations. The limitations of using only generic visual patterns are also clear in the synthesized images (c) which are obtained by stochastic sampling from the generative models after the parameters have been estimated by DDMCMC. The right panels (d) show the segmentations obtained by human subjects who, by contrast to the algorithm, appear to use object specific knowledge when doing the segmentation (though they were not instructed to) [34]. We conclude that to achieve good segmentation on these types of images requires combining segmentation with object detection and recognition.

(i) remove false alarms, (ii) amend missing objects by global context information, and (iii) reconcile conflicting (overlapping) explanations through model comparison. In this paper, we impose such processes by using generative models for the entire image.

As we will show, our image parsing algorithm is able to integrate discriminative and generative methods so as to take advantage of their complementary strengths. Moreover, we can couple modules such as segmentation and object detection by our choice of the set of visual patterns used to parse the image. In this paper, we focus on two types of patterns: – generic visual patterns for low/middle level vision, such as texture and shading, and object patterns at high level vision, such as frontal human faces and text.

These two types of patterns illustrate different ways in which the parsing graph can be constructed (see Figure 20 and the related discussion). The object patterns (face and text) have comparatively little variability so they can often be effectively detected as a whole by bottom-up tests and their parts can be located subsequently. Thus their parsing sub-graphs can be constructed in a “decompositional” manner from whole to parts. By contrast, a generic texture region

has arbitrary shape and its intensity pattern has high entropy. Detecting such a region by bottom-up tests will require an enormous number of tests to deal with all this variability, and so will be computationally impractical. Instead, the parsing subgraphs should be built by grouping small elements in a “compositional” manner [5].

We illustrate our algorithm on natural images of complex city scenes and give examples where image segmentation can be improved by allowing object specific knowledge to disambiguate low-level cues, and conversely object detection can be improved by using generic visual patterns to explain away shadows and occlusions.

This paper is structured as follows. In Section (2), we give an overview of the image parsing framework and discuss its theoretical background. Then in Section (3), we describe the parsing graph and the generative models used for generic visual patterns, text, and faces. In Section (4) we give the control structure of the image parsing algorithm. Section (5) gives details of the components of the algorithm. Section (6) shows how we combine AdaBoost with other tests to get proposals for detecting objects including text and faces. In Section (7) we present experimental results. Section (8) addresses some open problems in further developing the image parser as a general inference engine. We summarize the paper in Section (9).

2 Overview of Image Parsing Framework

2.1 Bottom-Up and Top-Down Processing

A major element of our work is to integrate discriminative and generative methods for inference. In the recent computer vision literature, top-down and bottom-up procedures can be broadly categorized into two popular inference paradigms – *generative* methods for “top-down” and *discriminative* methods for “bottom-up”, illustrated in Figure 3. From this perspective, integrating generative and discriminative models is equivalent to combining bottom-up and top-down processing.

The role of bottom-up and top-down processing in vision has been often discussed. There is growing evidence (see [44, 27]) that humans can perform high level scene and object categorization tasks as fast as low level texture discrimination and other so-called pre-attentive vision tasks. This suggests that humans can detect both low and high level visual patterns at early stages in visual processing. It contrasts with traditional bottom-up feedforward architectures [33] which start with edge detection, followed by segmentation/grouping, before proceeding to object recognition and

other high-level vision tasks. The experiments also relate to long standing conjectures about the role of the bottom-up/top-down loops in the visual cortical areas [37, 51], visual routines and pathways [50], the binding of visual cues [45], and neural network models such as the Helmholtz machine [14]. But although combining bottom-up and top-down processing is clearly important, there has not yet been a rigorous mathematical framework for how to achieve it.

In this paper, we unify generative and discriminative approaches by designing an DDMCMC algorithm which uses discriminative methods to perform rapid inference of the parameters of generative models. From a computer vision perspective, DDMCMC combines bottom-up processing, implemented by the discriminative models, together with top-down processing by the generative models. The rest of this section gives an overview of our approach.

2.2 Generative and Discriminative Methods

Generative methods specify how the image \mathbf{I} is generated from the scene representation $W \in \Omega$. It combines a prior $p(W)$ and a likelihood function $p(\mathbf{I}|W)$ to give a joint posterior probability $p(W|\mathbf{I})$. These can be expressed as probability probabilities on graphs, where the input image \mathbf{I} is represented on the leaf nodes and W denotes the remaining nodes and node attributes of the graph. The structure of the graph, for example the number of nodes, is unknown and must be estimated for each input image.

Inference can be performed by stochastic sampling W from the posterior:

$$W \sim p(W|\mathbf{I}) \propto p(\mathbf{I}|W)p(W). \quad (1)$$

This enables us to estimate $W^* = \arg \max P(W|\mathbf{I})$.³ But the dimension of the sample space Ω is very high and so standard sampling techniques are computationally expensive.

By contrast, discriminative methods are very fast to compute. They do not specify models for how the image is generated. Instead they give discriminative (conditional) probabilities $q(w_j|\text{Tst}_j(\mathbf{I}))$ for components $\{w_j\}$ of W based on a sequence of bottom-up tests $\text{Tst}_j(\mathbf{I})$ performed on the image. The tests are based on local image features $\{F_{j,n}(\mathbf{I})\}$ which can be computed from the image in a cascade manner (e.g. AdaBoost filters, see Section (6)),

$$\text{Tst}_j(\mathbf{I}) = (F_{j,1}(\mathbf{I}), F_{j,2}(\mathbf{I}), \dots, F_{j,n}(\mathbf{I})), \quad j = 1, 2, \dots, K. \quad (2)$$

³We are assuming that there are no known algorithms for estimating W^* directly.

The following theorem shows that the KL-divergence between the true marginal posterior $p(w_j|\mathbf{I})$ and the optimal discriminant approximation $q(w_j|\text{Tst}(\mathbf{I}))$ using test $\text{Tst}(\mathbf{I})$ will decrease monotonically as new tests are added⁴.

Theorem 1 *The information gained for a variable w by a new test $\text{Tst}_+(\mathbf{I})$ is the decrease of Kullback-Leibler divergence between $p(w|\mathbf{I})$ and its best discriminative estimate $q(w|\text{Tst}_t(\mathbf{I}))$ or the increase of mutual information between w and the tests.*

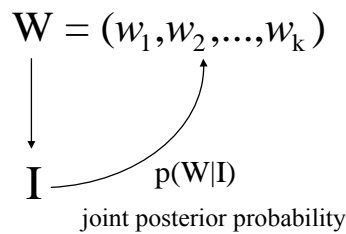
$$\begin{aligned} & E_{\mathbf{I}}[KL(p(w|\mathbf{I}) \parallel q(w|\text{Tst}(\mathbf{I})))] - E_{\mathbf{I}}[KL(p(w|\mathbf{I}) \parallel q(w|\text{Tst}(\mathbf{I}), \text{Tst}_+(\mathbf{I})))] \\ = & MI(w \parallel \text{Tst}, \text{Tst}_+) - MI(w \parallel \text{Tst}) = E_{\text{Tst}, \text{Tst}_+} KL(q(w \mid \text{Tst}_t, \text{Tst}_+) \parallel q(w \mid \text{Tst}_t)) \geq 0, \end{aligned}$$

where $E_{\mathbf{I}}$ is the expectation with respect to $P(\mathbf{I})$, and $E_{\text{Tst}, \text{Tst}_+}$ is the expectation with respect to the probability on the test responses $(\text{Tst}, \text{Tst}_+)$ induced by $P(\mathbf{I})$.

The decrease of the Kullback-Leibler divergence equals zero if and only if $\text{Tst}(\mathbf{I})$ are sufficient statistics with respect to w .

In practice discriminative methods, particularly standard computer vision algorithms – see subsection (4.1), will typically only use a small number of features for computational practicality. Also their discriminative probabilities $q(w_j|\text{Tst}(\mathbf{I}))$ will often not be optimal. Fortunately the image parsing algorithm in this paper only requires the discriminative probabilities $q(w_j|\text{Tst}(\mathbf{I}))$ to be rough approximations to $p(w_j|\mathbf{I})$.

Generative methods



Discriminative methods

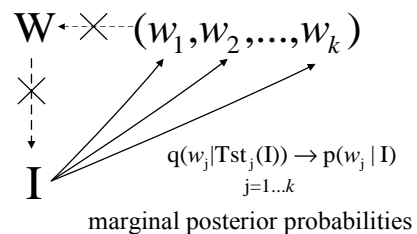


Figure 3: Comparison of two inference paradigms: Top-down *generative* methods versus bottom-up *discriminative* methods. The generative method specifies how the image \mathbf{I} can be synthesized from the scene representation W . By contrast, the discriminative methods are based by performing tests $\text{Tst}_j(\mathbf{I})$ and are not guaranteed to yield consistent solutions, see crosses explained in the text.

⁴The optimal approximation occurs when $q(w_j|\text{Tst}(\mathbf{I}))$ equals the probability $p(w_j|\text{Tst}(\mathbf{I}))$ induced by $P(\mathbf{I}|W)P(W)$.

The difference between discriminative and generative models is illustrated in Figure 3. Discriminative models are fast to compute and can be run in parallel because different components are computed independently (see arrows in Figure 3). But the components $\{w_i\}$ may not yield a consistent solution W and, moreover, W may not specify a consistent model for generating the observed image \mathbf{I} . These inconsistencies are indicated by the crosses in Figure 3. Generative models ensure consistency but require solving a difficult inference problem. It is an open problem whether discriminative models can be designed to infer the entire state W for the complicated generative models that we are dealing with. Mathematicians [6] have argued that this will not be practical and that discriminative models will always require additional post-processing.

2.3 Markov Chain kernels and sub-kernels

Formally, our DDMCMC image parsing algorithm simulates a Markov chain $\mathcal{MC} = \langle \Omega, \nu, \mathcal{K} \rangle$ with kernel \mathcal{K} in space Ω and with probability ν for the starting state. An element $W \in \Omega$ is a parsing graph. We let the set of parsing graphs Ω be finite as images have finite pixels and grey levels.

We proceed by defining a set of moves for reconfiguring the graph. These include moves to: (i) create nodes, (ii) delete nodes, and (iii) change node attributes. We specify stochastic dynamics for these moves in terms of transition kernels⁵.

For each move we define a Markov Chain sub-kernel by a transition matrix $\mathcal{K}_a(W'|W : \mathbf{I})$ with $a \in \mathcal{A}$ being an index. This represents the probability that the system makes a transition from state W to state W' when sub-kernel a is applied (i.e. $\sum_{W'} \mathcal{K}_a(W'|W : \mathbf{I}) = 1, \forall W$). Kernels which alter the graph structure are grouped into reversible pairs. For example, the sub-kernel for node creation $\mathcal{K}_{a,r}(W'|W : \mathbf{I})$ is paired with the sub-kernel for node deletion $\mathcal{K}_{a,l}(W'|W : \mathbf{I})$. This can be combined into a paired sub-kernel $\mathcal{K}_a = \rho_{ar}\mathcal{K}_{a,r}(W'|W : \mathbf{I}) + \rho_{al}\mathcal{K}_{a,l}(W'|W : \mathbf{I})$ ($\rho_{ar} + \rho_{al} = 1$). This pairing ensures that $\mathcal{K}_a(W'|W : \mathbf{I}) = 0$ if, and only if, $\mathcal{K}_a(W|W' : \mathbf{I}) = 0$ for all states $W, W' \in \Omega$. The sub-kernels (after pairing) are constructed to obey the detailed balance condition:

$$p(W|\mathbf{I})\mathcal{K}_a(W'|W : \mathbf{I}) = p(W'|\mathbf{I})\mathcal{K}_a(W|W' : \mathbf{I}). \quad (3)$$

⁵We choose stochastic dynamics because the Markov chain probability is guaranteed to converge to the posterior $P(W|\mathbf{I})$. The complexity of the problem means that deterministic algorithms for implementing these moves risk getting stuck in local minima.

The full transition kernel is expressed as:

$$\mathcal{K}(W'|W : \mathbf{I}) = \sum_a \rho(a : \mathbf{I}) \mathcal{K}_a(W'|W : \mathbf{I}), \quad \sum_a \rho(a : \mathbf{I}) = 1, \quad \rho(a : \mathbf{I}) > 0. \quad (4)$$

To implement this kernel, at each time step the algorithm selects the choice of move with probability $\rho(a : \mathbf{I})$ for move a , and then uses kernel $\mathcal{K}_a(W'|W; \mathbf{I})$ to select the transition from state W to state W' . Note that both probabilities $\rho(a : \mathbf{I})$ and $\mathcal{K}_a(W'|W; \mathbf{I})$ depend on the input image \mathbf{I} . This distinguishes our DDMCMC methods from conventional MCMC computing [28, 7].

The full kernel obeys detailed balance, equation (3), because all the sub-kernels do. It will also be ergodic, provided the set of moves is sufficient (i.e. so that we can transition between any two states $W, W' \in \Omega$ using these moves). These two conditions ensure that $p(W|\mathbf{I})$ is the invariant (target) probability of the Markov Chain [7] in the finite space Ω .

Applying the kernel $\mathcal{K}_{a(t)}$ updates the Markov chain state probability $\mu_t(W)$ at step t to $\mu_{t+1}(W')$ at $t + 1$,⁶:

$$\mu_{t+1}(W') = \sum_W \mathcal{K}_{a(t)}(W'|W : \mathbf{I}) \mu_t(W). \quad (5)$$

In summary, the DDMCMC image parser simulates a Markov chain \mathcal{MC} with a unique invariant probability $p(W|\mathbf{I})$. At time t , the Markov chain state (i.e. the parse graph) W follows a probability μ_t which is the product of the sub-kernels selected up to time t ,

$$W \sim \mu_t(W) = \nu(W_o) \cdot [\mathcal{K}_{a(1)} \circ \mathcal{K}_{a(2)} \circ \dots \circ \mathcal{K}_{a(t)}](W_o, W) \longrightarrow p(W|\mathbf{I}). \quad (6)$$

where $a(t)$ indexes the sub-kernel selected at time t . As the time t increases, $\mu_t(W)$ approaches the posterior $p(W|\mathbf{I})$ monotonically [7] at a geometric rate [15]. The following convergence theorem is useful for image parsing because it helps quantify the effectiveness of the different sub-kernels.

Theorem 2 *The Kullback-Leibler divergence between the posterior $p(W|\mathbf{I})$ and the Markov chain state probability decreases monotonically when a sub-kernel $\mathcal{K}_{a(t)}, \forall a(t) \in \mathcal{A}$ is applied,*

$$KL(p(W|\mathbf{I}) || \mu_t(W)) - KL(p(W|\mathbf{I}) || \mu_{t+1}(W)) \geq 0 \quad (7)$$

The decrease of KL-divergence is strictly positive and is equal to zero only after the Markov chain becomes stationary, i.e. $\mu = p$.

⁶Algorithms like belief propagation [55] can be derived as approximations to this update equation by using a Gibbs sampler and making independence assumptions.

[Proof] See Appendix A.

The theorem is related to the second law of thermodynamics [13], and its proof makes use of the detailed balance equation (3). This KL divergence gives a measure of the “power” of each sub-kernel $\mathcal{K}_{a(t)}$ and so it suggests an efficient mechanism for selecting the sub-kernels at each time step, see Section (8). By contrast, classic convergence analysis (see Appendix B) show that the convergence of the Markov Chain is exponentially fast, but does not give measures of power of sub-kernels.

2.4 DDMCMC and Proposal Probabilities

We now describe how to design the sub-kernels using proposal probabilities and discriminative models. This is at the heart of DDMCMC.

Each sub-kernel⁷ is designed to be of Metropolis-Hastings form [35, 24]:

$$\mathcal{K}_a(W'|W : \mathbf{I}) = Q_a(W'|W : \text{Tst}_a(\mathbf{I})) \min\left\{1, \frac{p(W'|\mathbf{I})Q_a(W|W' : \text{Tst}_a(\mathbf{I}))}{p(W|\mathbf{I})Q_a(W'|W : \text{Tst}_a(\mathbf{I}))}\right\}, \quad W' \neq W \quad (8)$$

where a transition from W to W' is proposed (stochastically) by the proposal probability $Q_a(W'|W : \text{Tst}_a(\mathbf{I}))$ and accepted (stochastically) by the acceptance probability:

$$\alpha(W'|W : \mathbf{I}) = \min\left\{1, \frac{p(W'|\mathbf{I})Q_a(W|W' : \text{Tst}_a(\mathbf{I}))}{p(W|\mathbf{I})Q_a(W'|W : \text{Tst}_a(\mathbf{I}))}\right\}. \quad (9)$$

Metropolis-Hastings ensures that the sub-kernels obey detailed balance (after pairing).

The proposal probability $Q_a(W'|W : \text{Tst}_a(\mathbf{I}))$ is a product (factorized) of some discriminative probabilities $q(w_j|\text{Tst}_j(\mathbf{I}))$ for the respective elements w_j changed in the move between W and W' (see later section). $\text{Tst}_a(\mathbf{I})$ is the set of bottom-up tests used in the proposal probabilities $Q_a(W'|W : \text{Tst}_a(\mathbf{I}))$ and $Q_a(W|W' : \text{Tst}_a(\mathbf{I}))$. The proposal probabilities must be fast to compute (because they should be evaluated for all the possible state W' that sub-kernel a can reach) and they should propose transitions to states W' where the posterior $p(W'|\mathbf{I})$ is likely to be high. The acceptance probabilities are more computationally expensive, because of their dependence on $p(W'|\mathbf{I})$, but they only need to be evaluated for the proposed state.

The design of the proposals is a trade-off. Ideally the proposals would be sampled from the posterior $p(W'|\mathbf{I})$, but this is impractical. Instead the trade-off requires: (i) the possibility of

⁷Except for one that evolves region boundaries.

making large moves in Ω at each time step, (ii) the proposals should encourage moves to states with high posterior probability, and (iii) the proposals must be fast to compute.

More formally, we define the scope $\Omega_a(W) = \{W' \in \Omega : \mathcal{K}_a(W'|W : \mathbf{I}) > 0\}$ to be the set of states which can be reached from W in one time step using sub-kernel a . We want the scope $S_a(W)$ to be large so that we can make large moves in the space Ω at each time step (i.e. jump towards the solution and not crawl). The scope should also, if possible, include states W' with high posterior $p(W'|\mathbf{I})$ (i.e. it is not enough for the scope to be large, it should also be in the right part of Ω).

The proposals $Q_a(W'|W : \text{Tst}_a(\mathbf{I}))$ should be chosen so as to approximate

$$\frac{p(W'|\mathbf{I})}{\sum_{W'' \in \Omega_a(W)} p(W''|\mathbf{I})} \text{ if } W' \in \Omega_a(W), \quad = 0, \text{ otherwise.} \quad (10)$$

The proposals will be functions of the discriminative models for the components of W' and of the generative models for the current state W (because it is computationally cheap to evaluate the generative models for the current state). The details of the model $p(W|\mathbf{I})$ will determine the form of the proposals and how large we can make the scope while keeping the proposals easy to compute and able to approximate equation (10). See the detailed examples given in Section (5).

This description gives the bare bones of DDMCMC. We refer to [47] for a more sophisticated discussion of these issues from an MCMC perspective. In the discussion section, we describe strategies to improve DDMCMC. Preliminary theoretical results for the convergence of DDMCMC are encouraging for a special case (see Appendix C).

Finally, in Appendix D, we address the important practical issue of how to maintain detailed balance when there are multiple routes to transition between two state W and W' . We describe two ways to do this and the trade-offs involved.

3 Generative models and Bayesian formulation

This section describes the parsing graph and the generative models used for our image parsing algorithm in this paper.

Figure 1 illustrates the general structure of a parsing graph. In this paper, we take a simplified two-layer-graph illustrated in Figure 4, which is fully specified in a generative sense. The top node (“root”) of the graph represents the whole scene (with a label). It has K *intermediate nodes* for the visual patterns (face, text, texture, and shading). Each visual pattern has a number of pixels at

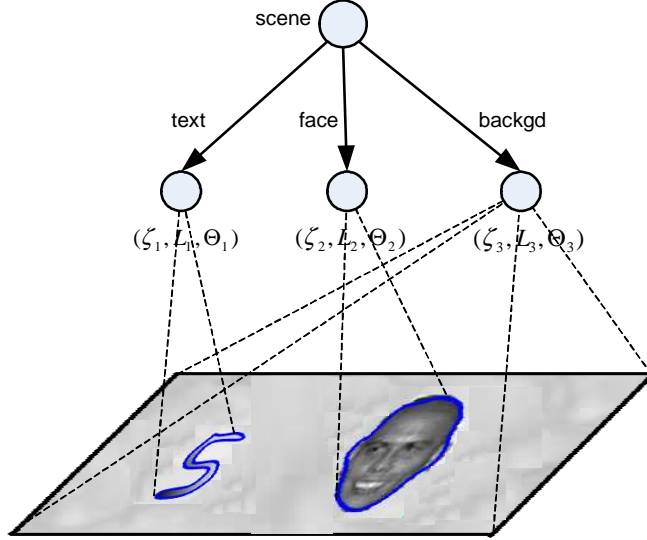


Figure 4: Abstract representation of the parsing graph used in this paper. The intermediate nodes represent the visual patterns. Their child nodes correspond to the pixels in the image.

the bottom (“leaves”). In this graph no horizontal connections are considered between the visual patterns except that they share boundaries and form a partition of the image lattice.

The number K of intermediate nodes is a random variable, and each node $i = 1, \dots, K$ has a set of attributes (L_i, ζ_i, Θ_i) defined as follows. L_i is the shape descriptor and determines the region $R_i = R(L_i)$ of the image pixels covered by the visual pattern of the intermediate node. Conceptually, the pixels within R_i are child nodes of the intermediate node i . (Regions may contain holes, in which case the shape descriptor will have internal and external boundaries). The remaining attribute variables (ζ_i, Θ_i) specify the probability models $p(\mathbf{I}_{R(L_i)} | \zeta_i, L_i, \Theta_i)$ for generating the sub-image $\mathbf{I}_{R(L_i)}$ in region $R(L_i)$. The variables $\zeta_i \in \{1, \dots, 66\}$ indicate the visual pattern type (3 types of generic visual patterns, 1 face pattern, and 62 text character patterns), and Θ_i denotes the model parameters for the corresponding visual pattern (details are given in the following sections). The complete scene description can be summarized by:

$$W = (K, \{(\zeta_i, L_i, \Theta_i) : i = 1, 2, \dots, K\}).$$

The shape descriptors $\{L_i : i = 1, \dots, K\}$ are required to be consistent so that each pixel in the image is a child of one, and only one, of the intermediate nodes. The shape descriptors must provide a partition of the image lattice $\Lambda = \{(m, n) : 1 \leq m \leq \text{Height}(\mathbf{I}), 1 \leq n \leq \text{Width}(\mathbf{I})\}$ and

hence satisfy the condition

$$\Lambda = \cup_{i=1}^K R(L_i), \quad R(L_i) \cap R(L_j) = \emptyset, \quad \forall i \neq j.$$

The generation process from the scene description W to \mathbf{I} is governed by the likelihood function:

$$p(\mathbf{I}|W) = \prod_{i=1}^K p(\mathbf{I}_{R(L_i)}|\zeta_i, L_i, \Theta_i).$$

The prior probability $p(W)$ is defined by

$$p(W) = p(K) \prod_{i=1}^K p(L_i)p(\zeta_i|L_i)p(\Theta_i|\zeta_i).$$

Under the Bayesian formulation, parsing the image corresponds to computing the W^* that maximizes *a posteriori* probability over Ω , the solution space of W ,

$$W^* = \arg \max_{W \in \Omega} p(W|\mathbf{I}) = \arg \max_{W \in \Omega} p(\mathbf{I}|W)p(W). \quad (11)$$

It remains to specify the prior $p(W)$ and the likelihood function $p(\mathbf{I}|W)$. We set the prior terms $p(K)$ and $p(\Theta_i|\zeta_i)$ to be uniform probabilities. The term $p(\zeta_i|L_i)$ is used to penalize high model complexity and was estimated for the three generic visual patterns from training data in [46].

3.1 Shape models

We use two types of shape descriptor in this paper. The first is used to define shapes of generic visual patterns and faces. The second defines the shapes of text characters.

1. Shape descriptors for generic visual patterns and faces

In this case, the shape descriptor represents the boundary⁸ of the image region by a list of pixels $L_i = \partial R_i$. The prior is defined by:

$$p(L_i) \propto \exp\{-\gamma|R(L_i)|^\alpha - \lambda|L_i|\}. \quad (12)$$

In this paper, we set $\alpha = 0.9$. For computational reasons, we use this prior for face shapes though more complicated priors [11] can be applied.

2. Shape descriptors for text characters

We model text characters by 62 deformable templates corresponding to the ten digits and the twenty six letters in both upper and lower cases. These deformable templates are defined by

⁸The boundary can include an ‘‘internal boundary’’ if there is a hole inside the image region explained by a different visual pattern.

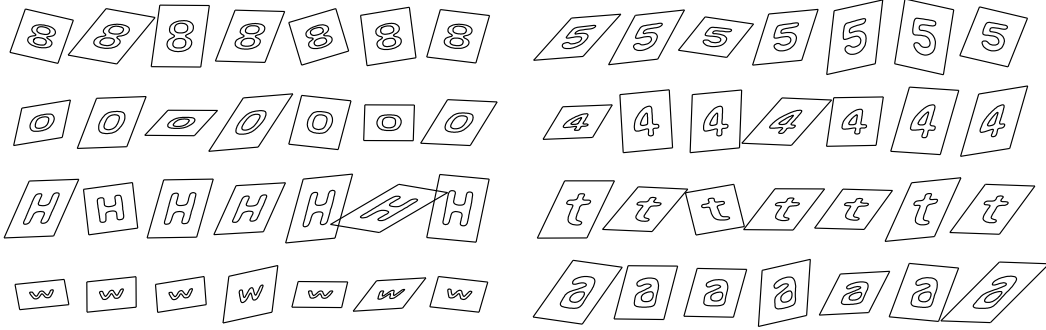


Figure 5: Random samples drawn from the shape descriptors for text characters.

62 prototype characters and a set of deformations. The prototypes are represented by an outer boundary and, at most, two inner boundaries. Each boundary is modeled by a B-spline using twenty five control points. The prototype characters are indexed by $c_i \in \{1, \dots, 62\}$ and their control points are represented by a matrix $TP(c_i)$.

We now define two types of deformations on the templates. One is a global affine transformation, and the other is a local elastic deformation. First we allow the letters to be deformed by an affine transform M_i . We put a prior $p(M_i)$ to penalize severe rotation and distortion. This is obtained by decomposing M_i as:

$$M_i = \begin{pmatrix} \sigma_x & 0 \\ 0 & \sigma_y \end{pmatrix} \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} 1 & h \\ 0 & 1 \end{pmatrix}.$$

where θ is the rotation angle, σ_x and σ_y denote scaling, and h is for shearing. The prior on M_i is

$$p(M_i) \propto \exp\{-a|\theta|^2 + b(\frac{\sigma_x}{\sigma_y} + \frac{\sigma_y}{\sigma_x})^2 + ch^2\},$$

where a, b, c are parameters.

Next, we allow local deformations by adjusting the positions of the B-spline control points. For a digit/letter c_i and affine transform M_i , the contour points of the template are given by $G_{TP}(M_i, c_i) = U \times M_s \times M_i \times TP(c_i)$. Similarly the contour points on the shape with control points S_i are given by $G_S(M_i, c_i) = U \times M_s \times S_i$ (U and M_s are the B-Spline matrices). We define a probability distribution $p(S_i|M_i, c_i)$ for the elastic deformation given by S_i ,

$$p(S_i|M_i, c_i) \propto \exp\{-\gamma|R(L_i)|^\alpha - D(G_S(M_i, c_i)||G_{TP}(M_i, c_i))\},$$

where $D(G_S(M_i, c_i)||G_{TP}(M_i, c_i))$ is the overall distance between contour template and the deformed contour (these deformations are small so the correspondence between points on the curves

can be obtained by nearest neighbor matches, see [48] for how we can refine this). Figure 5 shows some samples drawn from the above model.

In summary, each deformable template is indexed by $c_i \in \{1..62\}$ and has a shape descriptor:

$$L_i = (c_i, M_i, S_i),$$

The prior distribution on L_i is specified by:

$$p(L_i) = p(c_i)p(M_i)p(S_i|M_i, c_i).$$

Here $p(c_i)$ is a uniform distribution on all the digits and letters (we do not place a prior distribution on text strings, though it is possible to do so [25]).

3.2 Generative intensity models

We use four families of generative intensity models for describing intensity patterns of (approximately) constant intensity, clutter/texture, shading, and face. The first three are similar as those defined in [46].

1. Constant intensity model $\zeta = 1$:

This assumes that pixel intensities in a region R are subject to independently and identically distributed (iid) Gaussian distribution,

$$p_1(\mathbf{I}_{R(L)}|\zeta = 1, L, \Theta) = \prod_{v \in R(L)} G(\mathbf{I}_v - \mu; \sigma^2), \quad \Theta = (\mu, \sigma)$$

2. Clutter/texture model $\zeta = 2$:

This is a non-parametric intensity histogram $h()$ discretized to take G values (i.e. is expressed as a vector (h_1, h_2, \dots, h_G)). Let n_j be the number of pixels in $R(L)$ with intensity value j .

$$p_2(\mathbf{I}_{R(L)}|\zeta = 2, L, \Theta) = \prod_{v \in R(L)} h(\mathbf{I}_v) = \prod_{j=1}^G h_j^{n_j}, \quad \Theta = (h_1, h_2, \dots, h_G).$$

3. Shading model $\zeta = 3$ and $\zeta = 5, \dots, 66$:

This family of models are used to describe generic shading patterns, and text characters. We use a quadratic form

$$J(x, y; \Theta) = ax^2 + bxy + cy^2 + dx + ey + f,$$

with parameters $\Theta = (a, b, c, d, e, f, \sigma)$. Therefore, the generative model for pixel (x, y) is

$$p_3(\mathbf{I}_R(L)|\zeta \in \{3, (5, \dots, 66)\}, L, \Theta) = \prod_{v \in R(L)} G(\mathbf{I}_v - J_v; \sigma^2), \quad \Theta = (a, b, c, d, e, f, \sigma).$$

4. The PCA face model $\zeta = 4$:



Figure 6: Random samples drawn from the PCA face model.

The generative model for faces is simpler and uses Principal Component Analysis (PCA) to obtain representations of the faces in terms of principal components $\{B_i\}$ and covariances Σ . Lower level features, also modeled by PCA, can be added[36]. Figure 6 shows some faces sampled from the PCA model. We also add other features such as the occlusion process, as described in Hallinan et al [22].

$$p_4(\mathbf{I}_R(L)|\zeta = 4, L, \Theta) = G(\mathbf{I}_R(L) - \sum_i \lambda_i B_i; \Sigma), \quad \Theta = (\lambda_1, \dots, \lambda_n, \Sigma).$$

4 Overview of the Algorithm

This section gives the control structure of an image parsing algorithm based on the strategy described in section (2), and Figure 8 shows the diagram. Our algorithm must construct the parse graph on the fly and to estimate the scene interpretation W .

Figure 7 illustrates how the algorithm selects the Markov chain moves (dynamics or sub-kernels) to search through the space of possible parse graphs of the image by altering the graph structure (by deleting or adding nodes) and by changing the node attributes. An equivalent way of visualizing

the algorithm is in terms of a search through the solution space Ω , see [46, 47] for more details of this viewpoint.

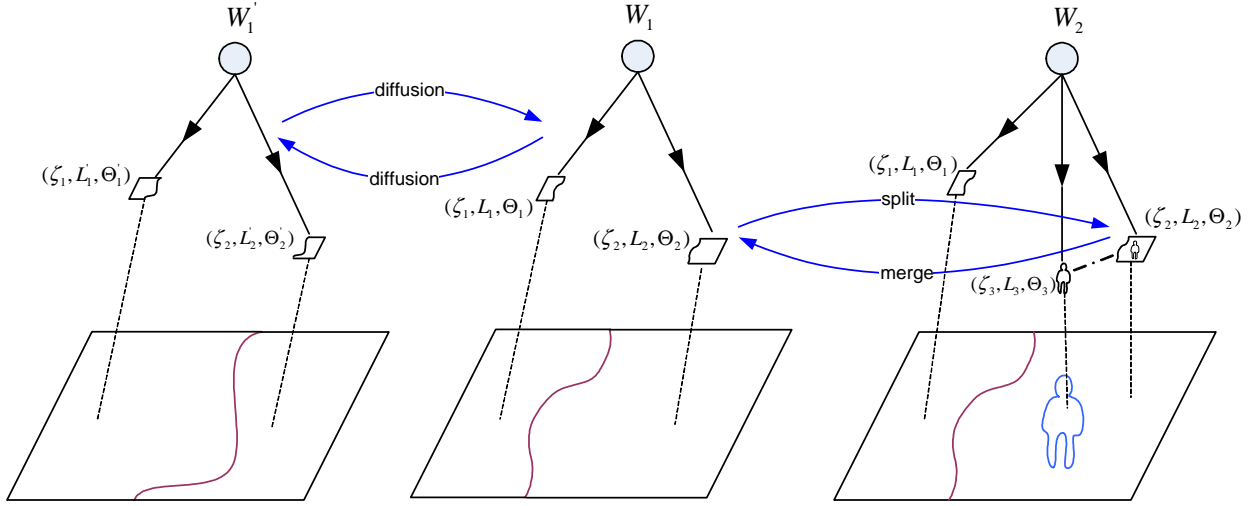


Figure 7: Examples of Markov chain dynamics that change the graph structure or the node attributes of the graph giving rise to different ways to parse the image.

We first define the set of moves to reconfigure the graph. These are: (i) birth or death of face nodes, (ii) birth or death of text characters, (iii) splitting or merging of regions, (iv) switching node attributes (region type ζ_i and model parameters Θ_i), (v) boundary evolution (altering the shape descriptors L_i of nodes with adjacent regions). These moves are implemented by sub-kernels. The first four moves are reversible jumps [21], and will be implemented by the Metropolis-Hastings equation (8). The fifth move, boundary evolution, is implemented by a stochastic partial differential equation.

The sub-kernels for these moves require proposal probabilities driven by elementary discriminative methods, which we review in the next subsection. The proposal probabilities are designed using the criteria in subsection (2.4), and full details are given in Section (5).

The control structure of the algorithm is described in Section (4.2). The full transition kernel for the image parser is built by combining the sub-kernels, as described in subsection (2.3) and Figure 8. The algorithm proceeds (stochastically) by selecting a sub-kernel, selecting where in the graph to apply it, and then deciding whether or not to accept the operation.

4.1 Discriminative Methods

The discriminative methods give approximate posterior probabilities $q(w_j|\text{Tst}_j(\mathbf{I}))$ for the elementary components w_j of W . For computational efficiency, these probabilities are based only on a small number of simple tests $\text{Tst}_j(\mathbf{I})$.

We briefly overview and classify the discriminative methods used in our implementation. Section (5) shows how these discriminative methods are composed, see crosses in Figure 8, to give proposals for making moves in the parsing graph.

1. Edge Cues. These cues are based on edge detectors [9],[8],[26]. They are used to give proposals for region boundaries (i.e. the shape descriptor attributes of the nodes). Specifically, we run the Canny detector at three scales followed by edge linking to give partitions of the image lattice. This gives a finite list of candidate partitions which are assigned weights, see section (5.2.3) and [46]. The discriminative probability is represented by this weighted list of particles. In principle, statistical edge detectors [26] would be preferable to Canny because they give discriminative probabilities $q(w_j|\text{Tst}_j(\mathbf{I}))$ learnt from training data.

2. Binarization Cues. These cues are computed using a variant of Niblack’s algorithm [39]. They are used to propose boundaries for text characters (i.e. shape descriptors for text nodes), and will be used in conjunction with proposals for text detection. The binarization algorithm, and an example of its output, are given in Section (6). Like edge cues, the algorithm is run at different parameters settings and represents the discriminative probability by a weighted list of particles indicating candidate boundary locations.

3. Face Region Cues. These cues are learnt by a variant of AdaBoost [42],[52] which outputs discriminative probabilities [19], see Section (6). They propose the presence of faces in sub-regions of the image. These cues are combined with edge detection to propose the localization of faces in an image.

4. Text Region Cues. These cues are also learnt by a probabilistic version of AdaBoost, see Section (6). The algorithm is applied to image windows (at a range of scales). It outputs a discriminative probability for the presence of text in each window. Text region cues are combined with binarization to propose boundaries for text characters.

5. Shape Affinity Cues. These act on shape boundaries, produced by binarization, to propose text characters. They use shape context cues [4] and information features [48] to propose

matches between the shape boundaries and the deformable template models of text characters.

6. Region Affinity Cues. These are used to estimate whether two regions R_i, R_j are likely to have been generated by the same visual pattern family and model parameters. They use an affinity similarity measure [43] of the intensity properties $\mathbf{I}_{R_i}, \mathbf{I}_{R_j}$.

7. Model Parameter and Visual Pattern Family cues. These are used to propose model parameters and visual pattern family identity. They are based on clustering algorithms, such as mean-shift [12]. The clustering algorithms depend on the model types and are described in [46].

In our current implementation, we conduct all the bottom-up tests $\text{Tst}_j(\mathbf{I}), j = 1, 2, \dots, K$ at an early stage for all the discriminative models $q_j(w_j|\text{Tst}_j(\mathbf{I}))$, and they are then combined to form composite tests $\text{Tst}_a(\mathbf{I})$ for each subkernel \mathcal{K}_a in equations (8,9). It may be more efficient to perform these test as required, see discussion in section (8).

4.2 Control Structure of the Algorithm

The control strategy used by our image parser is illustrated in Figure 8. The image parser explores the space of parsing graphs by a Markov Chain Monte Carlo sampling algorithm. This algorithm uses a transition kernel \mathcal{K} which is composed of sub-kernels \mathcal{K}_a corresponding to different ways to reconfigure the parsing graph. These sub-kernels come in reversible pairs⁹ (e.g. birth and death) and are designed so that the target probability distribution of the kernel is the generative posterior $p(W|\mathbf{I})$. At each time step, a sub-kernel is selected stochastically. The sub-kernels use the Metropolis-Hasting sampling algorithm, see equation (8), which proceeds in two stages. First, it proposes a reconfiguration of the graph by sampling from a proposal probability. Then it accepts or rejects this reconfiguration by sampling the acceptance probability.

To summarize, we outline the algorithm below. At each time step, it specifies (stochastically) which move to select (i.e. which sub-kernel), where to apply it in the graph, and whether to accept the move. The probability to select moves $\rho(a : \mathbf{I})$ was first set to be independent of \mathbf{I} , but we got better performance by adapting it using discriminative cues to estimate the number of faces and text characters in the image (see details below). The choice of where to apply the move is specified (stochastically) by the sub-kernel. For some sub-kernels it is selected randomly and for others is chosen based on a *fitness factor* (see details in section (5)), which measures how well the current model fits the image data. Some annealing is required to start the algorithm because of the limited

⁹Except for the boundary evolution sub-kernel which will be described separately, see Section 5.1.

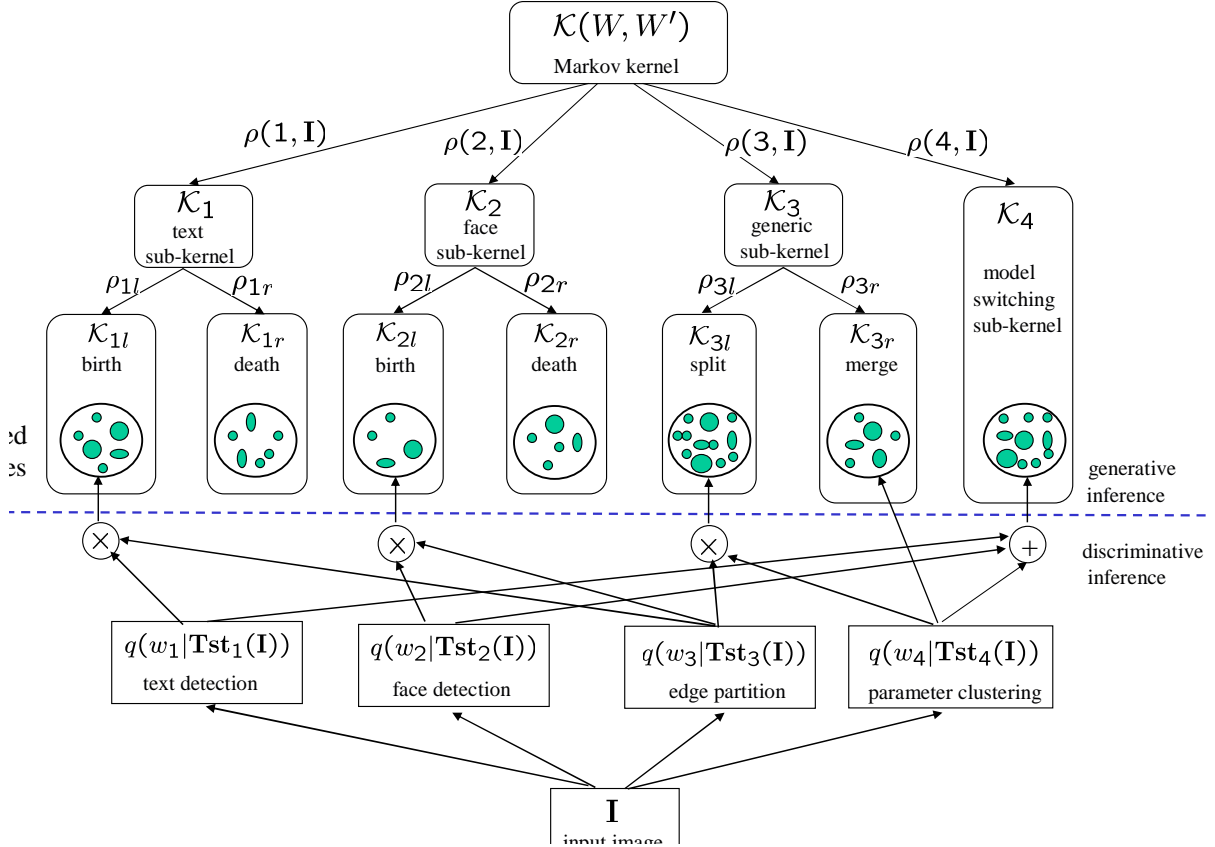


Figure 8: Integrating generative (top-down) and discriminative (bottom-up) methods for image parsing. This diagram illustrates the main points of the image parser. The dynamics are implemented by an ergodic Markov chain \mathcal{K} , whose invariant probability is the posterior $p(W|\mathbf{I})$, and which is composed of reversible sub-kernels \mathcal{K}_a for making different types of moves in the parse graph (e.g. giving birth to new nodes or merging nodes). At each time step the algorithm selects a sub-kernel stochastically. The selected sub-kernel proposes a specific move (e.g. to create or delete specific nodes) and this move is then evaluated and accepted stochastically, see equation (8). The proposals are based on both bottom-up (discriminative) and top-down (generative) processes, see subsection (2.4). The bottom-up processes compute discriminative probabilities $q(w_j|\text{Tst}_j(\mathbf{I}))$, $j = 1, 2, 3, 4$ from the input image \mathbf{I} based on feature tests $\text{Tst}_j(\mathbf{I})$. An additional sub-kernel for boundary evolution uses a stochastic partial differential equation will be described later.

scope of the moves in the current implementation (the need for annealing will be reduced if the compositional techniques described in [1] are used).

We improved the effectiveness of the algorithm by making the move selection adapt to the image (i.e. by making $\rho(a : \mathbf{I})$ depend on \mathbf{I}). In particular, we increased the probability of giving birth and death of faces and text, $\rho(1)$ and $\rho(2)$, if the bottom-up (AdaBoost) proposals suggested that there are many objects in the scene. For example, let $N(\mathbf{I})$ be the number of

proposals for faces or text above a threshold T_a . Then we modify the probabilities in the table by $\rho(a_1) \mapsto \{\rho(a_1) + kg(N(\mathbf{I}))\}/Z$, $\rho(a_2) \mapsto \{\rho(a_2) + kg(N)\}/Z$, $\rho(a_3) \mapsto \rho(a_3)/Z$, $\rho(a_4) \mapsto \rho(a_4)/Z$, where $g(x) = x$, $x \leq T_b$ $g(x) = T_b$, $x \geq T_b$ and $Z = 1 + 2k$ is chosen to normalize the probability.

The basic control strategy of the image parsing algorithm

1. Initialize W (e.g. by dividing the image into four regions), setting their shape descriptors, and assigning the remaining node attributes at random.
2. Set the temperature to be T_{init} .
3. Select the type a of move by sampling from a probability $\rho(a)$, with $\rho(1) = 0.2$ for faces, $\rho(2) = 0.2$ for text, $\rho(3) = 0.4$ for splitting and merging, $\rho(4) = 0.15$ for switching region model (type or model parameters), and $\rho(5) = 0.05$ for boundary evolution. This was modified slightly adaptively, see caption and text.
4. If the selected move is boundary evolution, then select adjacent regions (nodes) at random and apply stochastic steepest descent, see section (5.1).
5. If the jump moves are selected, then a new solution W' is randomly sampled as follows:
 - For the birth or death of a face, see section (5.2.2), we propose to create or delete a face. This includes a proposal for where in the image to do this.
 - For the birth or death of text, see section (5.2.1), we propose to create a text character or delete an existing one. This includes a proposal for where to do this.
 - For region splitting, see section (5.2.3), a region (node) is randomly chosen biased by its fitness factor. There are proposals for where to split it and for the attributes of the resulting two nodes.
 - For region merging, see section (5.2.3), two neighboring regions (nodes) are selected based on a proposal probability. There are proposals for the attributes of the resulting node.
 - For switching, see section (5.2.4), a region is selected randomly according to its fitness factor and a new region type and/or model parameters is proposed.
 - The full proposal probabilities, $Q(W|W : \mathbf{I})$ and $Q(W'|W : \mathbf{I})$ are computed.
 - The Metropolis-Hastings algorithm, equation (8), is applied to accept or reject the proposed move.
6. Reduce the temperature $T = 1 + T_{init} \times \exp(-t \times c|R|)$, where t is the current iteration step, c is a constant and $|R|$ is the size of the image.
7. Repeat the above steps and until the convergence criterion is satisfied (by reaching the maximum number of allowed steps or by lack of decrease of the negative log posterior).

5 The Markov Chain kernels

This section gives a detailed discussion of the individual Markov Chain kernel, their proposal probabilities, and their fitness factors.

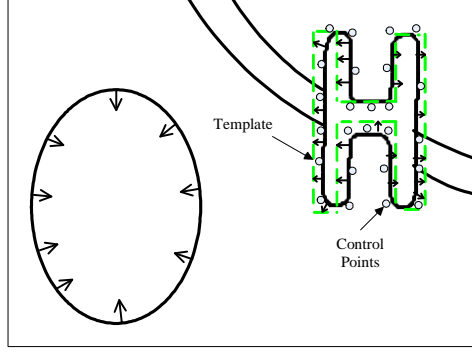


Figure 9: The evolution of the region boundaries is implemented by stochastic partial differential equations which are driven by models competing for ownership of the regions.

5.1 Boundary Evolution

These moves evolve the positions of the region boundaries but preserve the graph structure. They are implemented by a stochastic partial differential equation (Langevin equation) driven by Brownian noise and can be derived from a Markov Chain [20]. The deterministic component of the PDE is obtained by performing steepest descent on the negative log-posterior, as derived in [56].

We illustrate the approach by deriving the deterministic component of the PDE for the evolution of the boundary between a letter T_j and a generic visual pattern region \mathbf{R}_i . The boundary will be expressed in terms of the control points $\{S_m\}$ of the shape descriptor of the letter. Let v denote a point on the boundary, i.e. $v(s) = (x(s), y(s))$ on $\Gamma(s) = \partial R_i \cap \partial R_j$. The deterministic part of the evolution equation is obtained by taking the derivative of the negative log-posterior $-\log p(W|\mathbf{I})$ with respect to the control points.

The relevant parts of the negative log-posterior, see equation (11,3) are given by $E(\mathbf{R}_i)$ and $E(T_j)$ where:

$$E(\mathbf{R}_i) = \int \int_{R_i} -\log p(\mathbf{I}(x, y)|\theta_{\zeta_i}) dx dy + \gamma |R_i|^\alpha + \lambda |\partial R_i|.$$

and

$$E(T_j) = \int \int_{L_j} \log p(\mathbf{I}(x, y)|\theta_{\zeta_j}) dx dy + \gamma |R(L_j)|^\alpha - \log p(L_j).$$

Differentiating $E(R_i) + E(T_j)$ with respect to the control points $\{S_m\}$ yields the evolution PDE:

$$\frac{dS_m}{dt} = -\frac{\delta E(\mathbf{R}_i)}{\delta S_m} - \frac{\delta E(T_j)}{\delta S_m}$$

$$\begin{aligned}
&= \int \left[-\frac{\delta E(\mathbf{R}_i)}{\delta v} - \frac{\delta E(T_j)}{\delta v} \right] \frac{1}{|\mathbf{J}(s)|} ds \\
&= \int \mathbf{n}(v) \left[\log \frac{p(\mathbf{I}(v); \theta_{\zeta_i})}{p(\mathbf{I}(v); \theta_{\zeta_j})} + \alpha \gamma \left(\frac{1}{|D_j|^{1-\alpha}} - \frac{1}{|D_i|^{1-\alpha}} \right) - \lambda \kappa + D(G_{S_j}(s) || G_T(s)) \right] \frac{1}{|\mathbf{J}(s)|} ds,
\end{aligned}$$

where $\mathbf{J}(s)$ is the Jacobian matrix for the spline function. (Recall that $\alpha = 0.9$ in the implementation).

The log-likelihood ratio term $\log \frac{p(\mathbf{I}(v); \theta_{\zeta_i})}{p(\mathbf{I}(v); \theta_{\zeta_j})}$ implements the competition between the letter and the generic region models for ownership of the boundary pixels.

5.2 Markov chain sub-kernels

Changes in the graph structure are realized by Markov chain *jumps* implemented by four different sub-kernels.

5.2.1 Sub-kernel I: birth and death of text

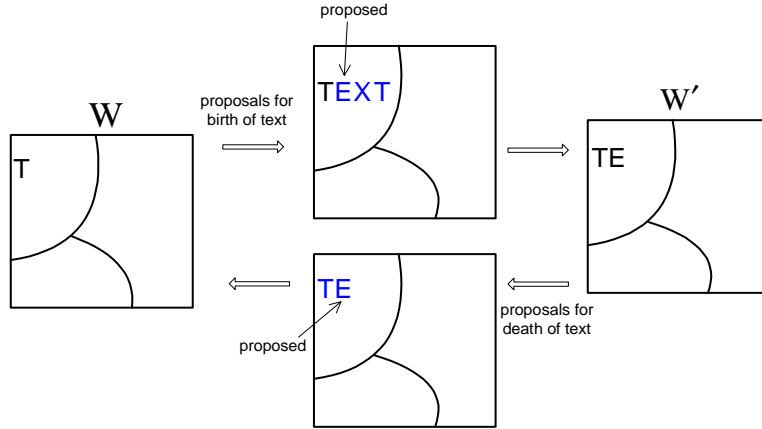


Figure 10: An example of the birth-death of text. State W consists of three generic regions and a character “T”. Proposals are computed for 3 candidate characters, “E”, “X”, and “T”, obtained by AdaBoost and binarization methods (see section (6.2)). One is selected, see arrow, which changes the state to W' . Conversely, there are 2 candidate in state W' and the one selected, see arrow, returns the system to state W .

This pair of jumps is used to create or delete text characters. We start with a parse graph W and transition into parse graph W' by creating a character. Conversely, we transition from W' back to W by deleting a character.

The proposals for creating and deleting text characters are designed to approximate the terms in equation (10). We obtain a list of candidate text character shapes by using AdaBoost to detect

text regions followed by binarization to detect candidate text character boundaries within text regions (see section (6.2)). This list is represented by a set of particles which are weighted by the similarity to the deformable templates for text characters (see below):

$$S_{1r}(W) = \{ (z_{1r}^{(\mu)}, \omega_{1r}^{(\mu)}) : \mu = 1, 2, \dots, N_{1r} \}.$$

Similarly, we specify another set of weighted particles for removing text characters:

$$S_{1l}(W') = \{ (z_{1l}^{(\nu)}, \omega_{1l}^{(\nu)}) : \nu = 1, 2, \dots, N_{1l} \}.$$

$\{z_{1r}^{(\mu)}\}$ and $\{z_{1l}^{(\nu)}\}$ represent the possible (discretized) shape positions and text character deformable templates for creating or removing text, and $\{\omega_{1r}^{(\mu)}\}$ and $\{\omega_{1l}^{(\nu)}\}$ are their corresponding weights. The particles are then used to compute proposal probabilities

$$\mathbf{Q}_{1r}(W'|W : \mathbf{I}) = \frac{\omega_{1r}(W')}{\sum_{\mu=1}^{N_{1r}} \omega_{1r}^{(\mu)}}, \quad \mathbf{Q}_{1l}(W|W', \mathbf{I}) = \frac{\omega_{1l}(W)}{\sum_{\nu=1}^{N_{1l}} \omega_{1l}^{(\nu)}}.$$

The weights $\omega_{1r}^{(\mu)}$ and $\omega_{1l}^{(\nu)}$ for creating new text characters are specified by shape affinity measures, such as shape contexts [4] and informative features [48]. For deleting text characters we calculate $\omega_{1l}^{(\nu)}$ directly from the likelihood and prior on the text character. Ideally these weights will approximate the ratios $\frac{p(W'|\mathbf{I})}{p(W|\mathbf{I})}$ and $\frac{p(W|\mathbf{I})}{p(W'|\mathbf{I})}$.

5.2.2 Sub-kernel II: birth and death of face

The sub-kernel for the birth and death of faces is very similar to the sub-kernel of birth and death of text. We use AdaBoost method discussed in sect. (6.2) to detect candidate faces. Face boundaries are obtained directly from using edge detection to give candidate face boundaries. The proposal probabilities are computed similarly to those for sub-kernel I.

5.2.3 Sub-kernel III: splitting and merging regions

This pair of jumps is used to create or delete nodes by splitting and merging regions (nodes). We start with a parse graph W and transition into parse graph W' by splitting node i into nodes j and k . Conversely, we transition back to W by merging nodes j and k into node i . The selection of which region i to split is based on a robust function on $p(\mathbf{I}_{R_i}|\zeta_i, L_i, \Theta_i)$ (i.e. the worse the model for region R_i fits the data, the more likely we are to split it). For merging, we use a region affinity measure [43] and propose merges between regions which have high affinity.

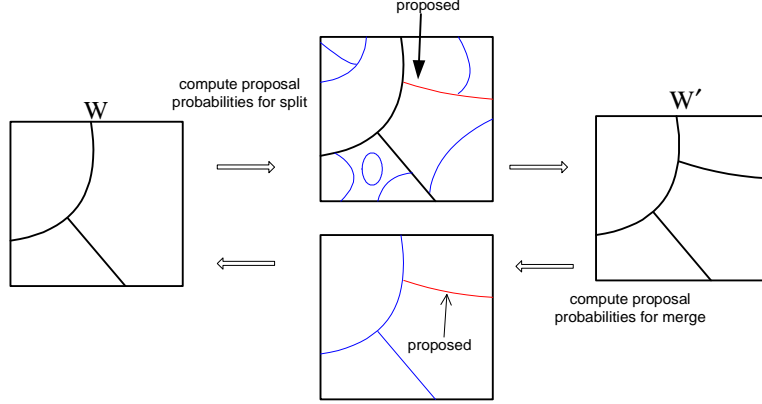


Figure 11: An example of the split-merge sub-kernel. State W consists of three regions and proposals are computed for 7 candidate splits. One is selected, see arrow, which changes the state to W' . Conversely, there are 5 candidate merges in state W' and the one selected, see arrow, returns the system to state W .

Formally, we define W, W' :

$$W = (K, (\zeta_k, L_k, \Theta_k), W_-) \Leftrightarrow W' = (K + 1, (\zeta_i, L_i, \Theta_i), (\zeta_j, L_j, \Theta_j), W_-)$$

where W_- denotes the attributes of the remaining $K - 1$ nodes in the graph.

We obtain proposals by seeking approximations to equation (10) as follows.

We first obtain three edge maps. These are given by Canny edge detectors [9] at different scales (see [46] for details). We use these edge maps to create a list of particles for splitting $S_{3r}(W)$. A list of particles for merging is denoted by $S_{3l}(W')$.

$$S_{3r}(W) = \{ (z_{3r}^{(\mu)}, \omega_{3r}^{(\mu)}) : \mu = 1, 2, \dots, N_{3r} \}, \quad S_{3l}(W') = \{ (z_{3l}^{(\nu)}, \omega_{3l}^{(\nu)}) : \nu = 1, 2, \dots, N_{3l} \}$$

where $\{z_{3r}^{(\mu)}\}$ and $\{z_{3l}^{(\nu)}\}$ represent the possible (discretized) positions for splitting and merging, and their weights $\{\omega_{3r}\}, \{\omega_{3l}\}$ will be defined shortly. In other words, we can *only* split a region i into regions j and k along a contour z_{3r}^μ (i.e. z_{3r}^μ forms the new boundary). Similarly we can only merge regions j and k into region i by deleting a boundary contour z_{3l}^ν . For example, Figure 11 shows 7 candidate sites for splitting W and 5 sites for merging W' .

We now define the weights $\{\omega_{3r}\}, \{\omega_{3l}\}$. These weights will be used to determine probabilities for the splits and merges by:

$$\mathbf{Q}_{3r}(W'|W : \mathbf{I}) = \frac{\omega_{3r}(W')}{\sum_{\mu=1}^{N_{3r}} \omega_{3r}^{(\mu)}}, \quad \mathbf{Q}_{3l}(W|W' : \mathbf{I}) = \frac{\omega_{3l}(W)}{\sum_{\nu=1}^{N_{3l}} \omega_{3l}^{(\nu)}}.$$

Again, we would like ω_{3r}^μ and ω_{3l}^ν to approximate the ratios $\frac{p(W|\mathbf{I})}{p(W'|\mathbf{I})}$ and $\frac{p(W'|\mathbf{I})}{p(W|\mathbf{I})}$ respectively. $\frac{p(W'|\mathbf{I})}{p(W|\mathbf{I})}$ is given by:

$$\frac{p(W'|\mathbf{I})}{p(W|\mathbf{I})} = \frac{p(\mathbf{I}_{R_i}|\zeta_i, L_i, \Theta_i)p(\mathbf{I}_{R_j}|\zeta_j, L_j, \Theta_j)}{p(\mathbf{I}_{R_k}|\zeta_k, L_k, \Theta_k)} \cdot \frac{p(\zeta_i, L_i, \Theta_i)p(\zeta_j, L_j, \Theta_j)}{p(\zeta_k, L_k, \Theta_k)} \cdot \frac{p(K+1)}{p(K)}$$

This is expensive to compute, so we approximate $\frac{p(W'|\mathbf{I})}{p(W|\mathbf{I})}$ and $\frac{p(W|\mathbf{I})}{p(W'|\mathbf{I})}$ by:

$$\omega_{3r}^{(\mu)} = \frac{q(R_i, R_j)}{p(\mathbf{I}_{R_k}|\zeta_k, L_k, \Theta_k)} \cdot \frac{[q(L_i)q(\zeta_i, \Theta_i)][q(L_j)q(\zeta_j, \Theta_j)]}{p(\zeta_k, L_k, \Theta_k)}. \quad (13)$$

$$\omega_{3l}^{(\nu)} = \frac{q(R_i, R_j)}{p(\mathbf{I}_{R_i}|\zeta_i, L_i, \Theta_i)p(\mathbf{I}_{R_j}|\zeta_j, L_j, \Theta_j)} \cdot \frac{q(L_k)q(\zeta_k, \Theta_k)}{p(\zeta_i, L_i, \Theta_i)p(\zeta_j, L_j, \Theta_j)}, \quad (14)$$

Where $q(R_i, R_j)$ is an *affinity measure* [43] of the similarity of the two regions R_i and R_j (it is a weighted sum of the intensity difference $|\bar{I}_i - \bar{I}_j|$ and the chi-squared difference between the intensity histograms), $q(L_i)$ is given by the priors on the shape descriptors, and $q(\zeta_i, \Theta_i)$ is obtained by clustering in parameter space (see [46]).

5.2.4 Jump II: Switching Node Attributes

These moves switch the attributes of a node i . This involves changing the region type ζ_i and the model parameters Θ_i .

The move transitions between two states:

$$W = ((\zeta_i, L_i, \Theta_i), W_-) \Leftrightarrow W' = ((\zeta'_i, L'_i, \Theta'_i), W_-)$$

The proposal, see equation (10), should approximate:

$$\frac{p(W'|\mathbf{I})}{p(W|\mathbf{I})} = \frac{p(\mathbf{I}_{R_i}|\zeta'_i, L'_i, \Theta'_i)p(\zeta'_i, L'_i, \Theta'_i)}{p(\mathbf{I}_{R_i}|\zeta_i, L_i, \Theta_i)p(\zeta_i, L_i, \Theta_i)}.$$

We approximate this by a weight $\omega_4^{(\mu)}$ given by

$$\omega_4^{(\mu)} = \frac{q(L'_i)q(\zeta'_i, \Theta'_i)}{p(\mathbf{I}_{R_i}|\zeta_i, L_i, \Theta_i)p(\zeta_i, L_i, \Theta_i)},$$

where $q(L'_i)q(\zeta'_i, \Theta'_i)$ are the same functions used in the split and merge moves. The proposal probability is the weight normalized in the candidate set, $\mathbf{Q}_4(W'|W : \mathbf{I}) = \frac{\omega_4(W')}{\sum_{\mu=1}^{N_4} \omega_4^{(\mu)}}$.

6 AdaBoost for discriminative probabilities for face and text

This section describes how we use AdaBoost techniques to compute discriminative probabilities for detecting faces and text (strings of letters). We also describe the binarization algorithm used to detect the boundaries of text characters.

6.1 Computing discriminative probabilities by Adaboost

The standard AdaBoost algorithm, for example for distinguishing faces from non-faces [52], learns a binary-valued *strong classifier* H_{Ada} by combining a set of n binary-valued “weak classifiers” or feature tests $\text{Tst}_{\text{Ada}}(\mathbf{I}) = (h_1(\mathbf{I}), \dots, h_n(\mathbf{I}))$ using a set of weights $\boldsymbol{\alpha}_{\text{Ada}} = (\alpha_1, \dots, \alpha_n)$ [18],

$$H_{\text{Ada}}(\text{Tst}_{\text{Ada}}(\mathbf{I})) = \text{sign}\left(\sum_{i=1}^n \alpha_i h_i(\mathbf{I})\right) = \text{sign} \langle \boldsymbol{\alpha}_{\text{Ada}}, \text{Tst}_{\text{Ada}}(\mathbf{I}) \rangle. \quad (15)$$

The features are selected from a pre-designed dictionary Δ_{Ada} . The selection of features and the tuning of weights are posed as a supervised learning problem. Given a set of labeled examples, $\{(\mathbf{I}_i, \ell_i) : i = 1, 2, \dots, M\}$ ($\ell_i = \pm 1$), AdaBoost learning can be formulated as greedily optimizing the following function [42]

$$(\boldsymbol{\alpha}_{\text{Ada}}^*, \text{Tst}_{\text{Ada}}^*) = \arg \min_{\text{Tst}_{\text{Ada}} \subset \Delta_{\text{Ada}}} \arg \min_{\boldsymbol{\alpha}_{\text{Ada}}} \sum_{i=1}^M \exp^{-\ell_i \langle \boldsymbol{\alpha}_{\text{Ada}}, \text{Tst}_{\text{Ada}}(\mathbf{I}_i) \rangle}. \quad (16)$$

To obtain discriminative probabilities we use a theorem [19] which states that the features and test learnt by AdaBoost give (asymptotically) posterior probabilities for the object labels (e.g. face or non-face). The AdaBoost strong classifier can be rederived as the log posterior ratio test.

Theorem 3 (Friedman et al 1998) *With sufficient training samples M and features n , AdaBoost learning selects the weights $\boldsymbol{\alpha}_{\text{Ada}}^*$ and tests $\text{Tst}_{\text{Ada}}^*$ to satisfy*

$$q(\ell = +1|\mathbf{I}) = \frac{e^{\langle \boldsymbol{\alpha}_{\text{Ada}}^*, \text{Tst}_{\text{Ada}}^*(\mathbf{I}) \rangle}}{e^{\langle \boldsymbol{\alpha}_{\text{Ada}}^*, \text{Tst}_{\text{Ada}}^*(\mathbf{I}) \rangle} + e^{-\langle \boldsymbol{\alpha}_{\text{Ada}}^*, \text{Tst}_{\text{Ada}}^*(\mathbf{I}) \rangle}}.$$

Moreover, the strong classifier converges asymptotically to the posterior probability ratio test

$$H_{\text{Ada}}(\text{Tst}_{\text{Ada}}(\mathbf{I})) = \text{sign}(\langle \boldsymbol{\alpha}_{\text{Ada}}, \text{Tst}_{\text{Ada}}(\mathbf{I}) \rangle) = \text{sign}\left(\frac{q(\ell = +1|\mathbf{I})}{q(\ell = -1|\mathbf{I})}\right).$$

In practice, the AdaBoost classifier is applied to windows in the image at different scales. Each window is evaluated as being face or non-face (or text versus non-text). For most images the posterior probabilities for faces or text are negligible for almost all parts of an image. So we use a cascade of tests [52, 54] which enables us to rapidly reject many windows by setting their marginal probabilities to be zero.

Of course, AdaBoost will only converge to approximations to the true posterior probabilities $p(\ell|\mathbf{I})$ because only a limited number of tests can be used (and there is only a limited amount of training data).

Note that AdaBoost is only one way to learn a posterior probability, see theorem (1). It has been found to be very effective for object patterns which have relatively rigid structures, such as faces and text (the shapes of letters are variable but the patterns of a sequence are fairly structured [10]).

6.2 AdaBoost Training

We used standard AdaBoost training methods [18, 19] combined with the cascade approach using asymmetric weighting[52, 54]. The cascade enables the algorithm to rule out most of the image as face, or text, locations with a few tests and allows computational resources to be concentrated on the more challenging parts of the images.



Figure 12: Some scenes from which the training text patches are extracted.

The AdaBoost for text was designed to detect text segments. Our test data was extracted by hand from 162 images of San Francisco, see Figure 12, and contained 561 text images, some of which can be seen in Figure 13. More than half of the images were taken by blind volunteers (which reduces bias). We divided each text image into several overlapping text segments with fixed width-to-height ration 2:1 (typically containing between two and three letters). A total of 7,000 text segments were used as the positive training set. The negative examples were obtained by a bootstrap process similar to Drucker et al [16]. First we selected negative examples by randomly sampling from windows in the image dataset. After training with these samples, we applied the AdaBoost algorithm at a range of scales to classify all windows in the training images. Those misclassified as text were then used as negative examples for the next stage of AdaBoost. The image regions most easily confused with text were vegetation, and repetitive structures such as railings or building facades. The features used for AdaBoost were image tests corresponding to the



a. Examples of text image from which we extracted text segments.



b. Examples of the faces in training.

Figure 13: Positive training examples for AdaBoost.

statistics of elementary filters. The features were chosen to detect properties of text segments that were relatively invariant to the shapes of the individual letters or digits. They included averaging the intensity within image windows, and statistics of the number of edges. We refer to [10] for more details.

Figure 14(a) shows some failure examples of Adaboost for text detection. These correspond to situations such as heavy shading, blurred images, isolated digits, vertical commercial signs, and non-standard fonts. They were not included in the training examples shown in Figure 13.

The AdaBoost posteriors for faces was trained in a similar way. This time we used Haar basis functions [52] as elementary features. We used the FERET [40] database for our positive examples, see Figure 13(b), and by allowing small rotation and translation transformation we had 5,000 positive examples. We used the same strategy as described above for text to obtain negative examples. Failure examples are shown in Figure 14.

In both cases, we evaluated the log posterior ratio test on testing datasets using a number of different thresholds (see [52]). In agreement with previous work on faces [52], AdaBoost gave very high performance with very few false positives and false negatives, see table (1). But these low error rates are slightly misleading because of the enormous number of windows in each image, see



a. Examples of difficult text



b. Examples of difficult faces

Figure 14: Some failure examples of text and faces that Adaboost misses. The text image failures (a) are challenging and were not included in our positive training examples. The face failures (b) include heavy shading, occlusion and facial features.

table (1). A small false positive rate may imply a large number of false positives for any regular image. By varying the threshold, we can either eliminate the false positives or the false negatives but not both at the same time. We illustrate this by showing the face regions and text regions proposed by AdaBoost in Figure 15. If we attempt classification by putting a threshold then we can only correctly detect all the faces and the text at the expense of false positives.

Object	False Positive	False Negative	Images	Subwindows
Face	65	26	162	355,960,040
Face	918	14	162	355,960,040
Face	7542	1	162	355,960,040
Text	118	27	35	20,183,316
Text	1879	5	35	20,183,316

Table 1: Performance of AdaBoost at different thresholds.

When Adaboost is integrated with the generic region models in the image parser, the generic region proposals can remove false positives and find text that Adaboost misses. For example, the '9' in the right panel of Figure 15 is not detected because our Adaboost algorithm was trained on text segments. Instead it is detected as a generic shading region and later recognized as a letter



Figure 15: The boxes show faces and text as detected by the AdaBoost log posterior ratio test with fixed threshold. Observe the false positives due to vegetation, tree structure, and random image patterns. It is impossible to select a threshold which has no false positives and false negatives for this image. As it is shown in our experiments later, the generative models will remove the false positives and also recover the missing text.

‘9’, see Figure 17. Some false positive text and faces in figure 15 are removed in Figures 17 and 19.

The AdaBoost algorithm for text needs to be supplemented with a binarization algorithm, described below, to determine text character location. This is followed by applying shape contexts [4] and informative features [48] to the binarization results to make proposals for the presence of specific letters and digits.

In many cases, see Figure 16, the results of binarization are so good that the letters and digits can be detected immediately (i.e. the proposals made by the binarization stage are automatically accepted). But this will not always be the case. We note that binarization gives far better results than alternatives such as edge detection [9].



Figure 16: Example of binarization on the detected text.

The binarization algorithm is a variant of one proposed by Niblack [39]. We binarize the image intensity using an adaptive thresholding based on a adaptive window size. Adaptive methods are needed because image windows containing text often have shading, shadow, and occlusion. Our binarization method determines the threshold $T_b(v)$ for each pixel v by the intensity distribution of its local window $r(v)$ (centered on v).

$$T_b(v) = \mu(\mathbf{I}_{r(v)}) + k \cdot \text{std}(\mathbf{I}_{r(v)}),$$

where $\mu(\mathbf{I}_{r(v)})$ and $\text{std}(\mathbf{I}_{r(v)})$ are the intensity mean and standard deviation within the local window. The size of the local window is selected to be the smallest possible window whose intensity variance is above a fixed threshold. The parameter $k = \pm 0.2$, where the \pm allows for cases where the foreground is brighter or darker than the background.

7 Experiments

The image parsing algorithm is applied to a number of outdoor/indoor images. The speed in PCs (Pentium IV) is comparable to segmentation methods such as normalized cuts [31] or the DDMCMC algorithm in [46]. It typically runs around 10-40 minutes. The main portion of the computing time is spent at segmenting generic regions and boundary diffusion [56].

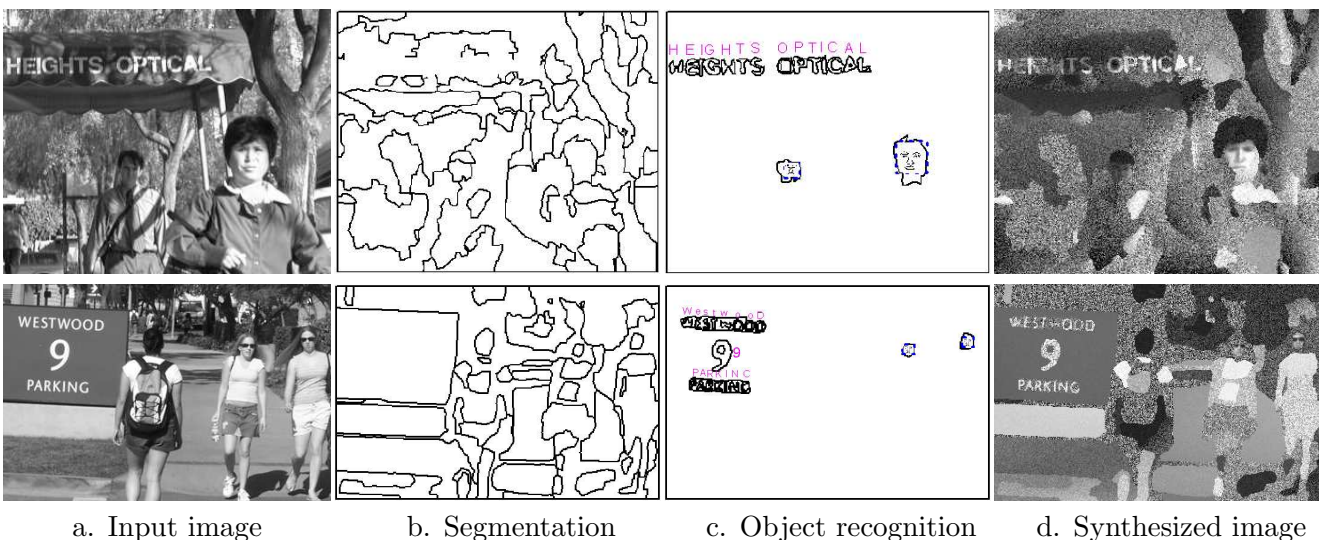


Figure 17: Results of segmentation and recognition on two images. The results are improved compare to the purely bottom-up (AdaBoost) results displayed in Figure 15.

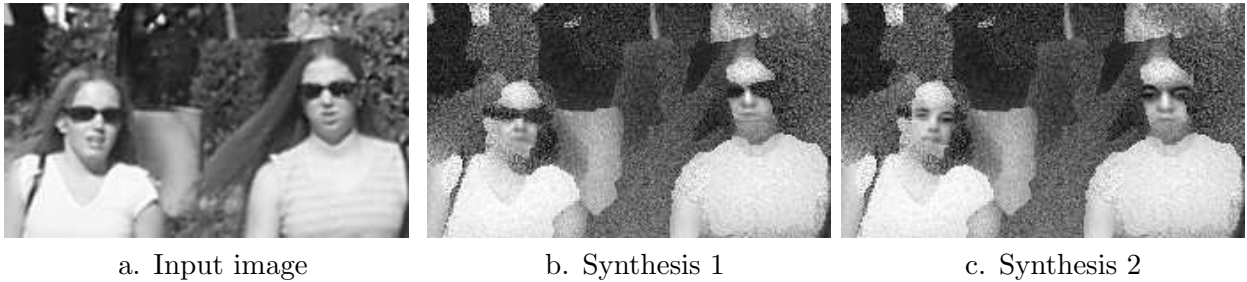


Figure 18: A close-up look of an image in Figure 17. The dark glasses are explained by the generic shading model and so the face model does not have to fit this part of the data. Otherwise the face model would have difficulty because it would try to fit the glasses to eyes. Standard AdaBoost only correctly classifies these faces at the expense of false positives, see Figure 15. We show two examples of synthesized faces, one (Synthesis 1) with the dark glasses (modelled by shading regions) and the other (Synthesis 2) with the dark glasses removed (i.e. using the generative face model to sample parts of the face (e.g. eyes) obscured by the dark glasses).

Figures 17, 18, and 19 show some challenging examples which have heavy clutter and shading effects. We present the results in two parts. One shows the segmentation boundaries for generic regions and objects, and the other shows the text and faces detected with text symbols to indicate text recognition, i.e. the letters are correctly read by the algorithm. Then we synthesize images sampled from the likelihood model $p(\mathbf{I}|W^*)$ where W^* is the parsing graph (the faces, text, regions parameters and boundaries) obtained by the parsing algorithm. The synthesized images are used to visualize the parsing graph W^* , i.e. the image content that the computer “understand”.

In the experiments, we observed that the face and text models improved the image segmentation results by comparison to our previous work [46] which only used generic region models. Conversely, the generic region models improve object detection by removing some false alarms and recovering objects which were not initially detected. We now discuss specific examples.

In Figure 15, we showed two images where the text and faces were detected purely bottom-up using AdaBoost. It is impossible to select a threshold so that our AdaBoost algorithm had no false positives or false negatives. To ensure no false negatives, apart from the ‘9’, we had to lower the threshold and admit false positives due to vegetation and heavy shadows (e.g. the shadow in the sign “HEIGHTS OPTICAL”).

The letter ‘9’ was not detected at any threshold. This is because our AdaBoost algorithm was trained to detect text segments, and so did not respond to a single digit.

By comparison, Figure 17 shows the image parsing results for these two images. We see that the

false alarms proposed by AdaBoost are removed because they are better explained by the generic region models. The generic shading models help object detection by explaining away the heavy shading on the text “HEIGHTS OPTICAL” and the dark glasses on the women, see Figure 18. Moreover, the missing digit ‘9’ is now correctly detected. The algorithm first detected it as a generic shading region and then reclassified as a digit using the sub-kernel that switches node attributes.

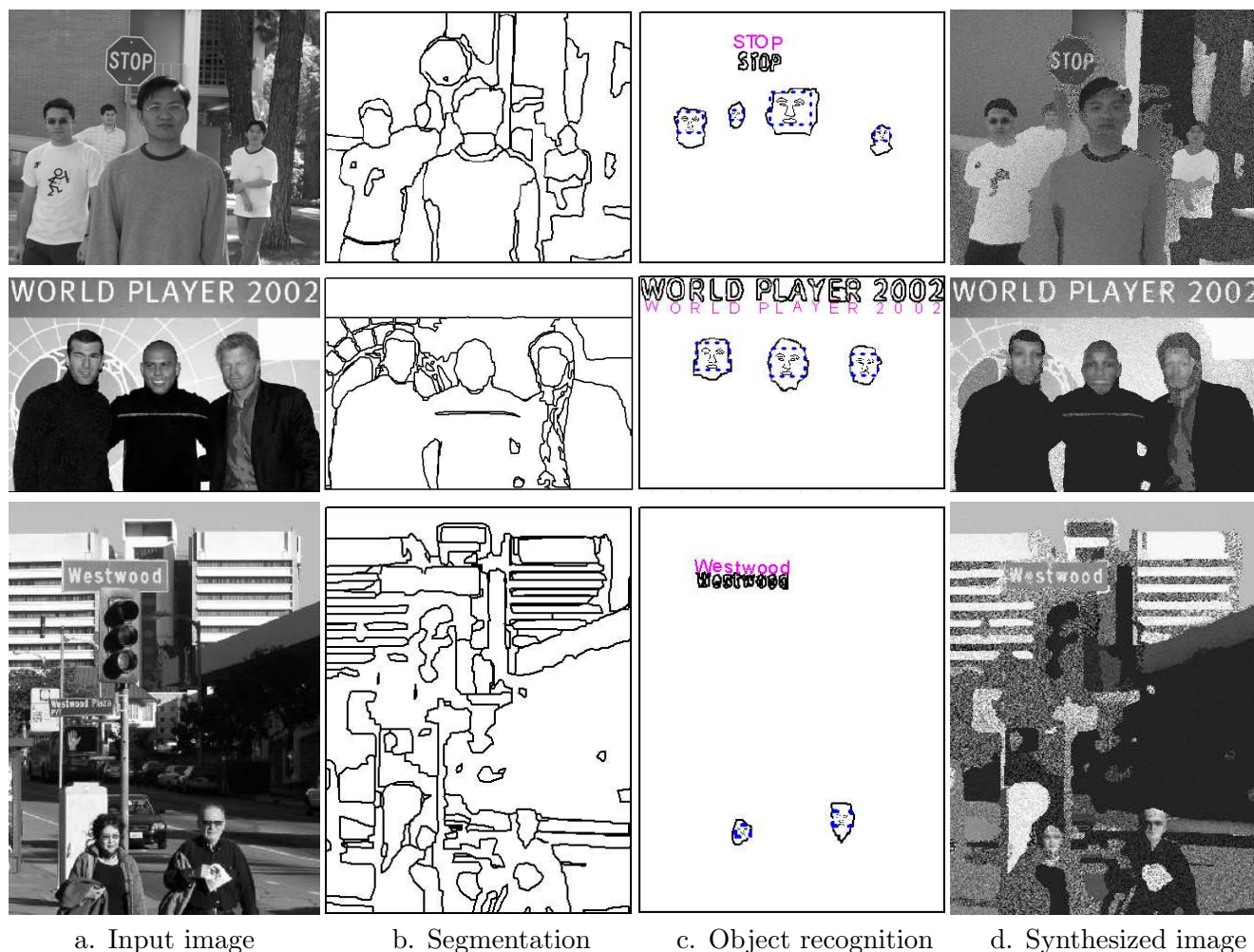


Figure 19: Results of segmentation and recognition on outdoor images. Observe the ability to detect faces and text at multiple scale.

The ability to synthesize the image from the parsing graph W^* is an advantage of the Bayesian approach. The synthesis helps illustrate the successes, and sometimes the weaknesses, of the generative models. Moreover, the synthesized images show how much information about the image has been captured by the models. In table (2), we give the number of variables used in our represen-

tation W^* and show that they are roughly proportional to the jpeg bytes. Most of the variables in W^* are used to represent points on the segmentation boundary, and at present they are counted independently. We could reduce the coding length of W^* substantially by encoding the boundary points effectively, for example, using spatial proximity. Image encoding is not the goal of our current work, however, and more sophisticated generative models would be needed to synthesize very realistic images.

Image	Stop	Soccer	Parking	Street	Westwood
jpg bytes	23,998	19,563	23,311	26,170	27,790
$ W^* $	4,886	3,971	5,013	6,346	9,687

Table 2: The number of variables in W^* for each image compared to the JPG bytes.

8 Discussion

In this section, we describe two challenging technical problems for image parsing. Our current work addresses these issues.

1. *Two mechanisms for constructing the parsing graph*

In the introduction to this paper we stated that the parsing graph can be constructed in compositional and decompositional modes. The compositional mode proceeds by grouping small elements while the decompositional approach involves detecting an object as a whole and then locating its parts, see Figure 20.

The compositional mode appears most effective for Figure 20(a). Detecting the cheetah by bottom-up tests, such as those learnt by AdaBoost, seems difficult owing to the large variability of shape and photometric properties of cheetahs. By contrast, it is quite practical using Swendsen-Wang Cuts [2] to segment the image and obtain the boundary of the cheetah using a bottom-up compositional approach and a parsing tree with multiple levels. The parsing graph is constructed starting with the pixels as leaves (there are 46,256 pixels in Figure 20(a)). The next level of the graph is obtained using local image texture similarities to construct graph nodes (113 of them) corresponding to “atomic regions” of the image. Then the algorithm constructs nodes (4 of them) for “texture regions” at the next level by grouping the atomic regions (i.e. each atomic region node will be the child of a texture region node). At each level, we compute a discriminative (proposal)

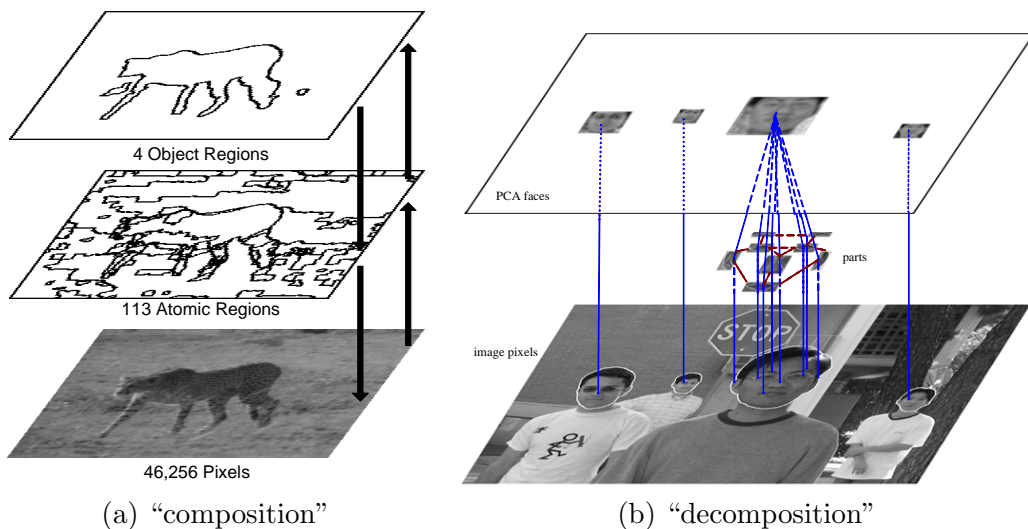


Figure 20: Two mechanisms for constructing the parsing graph. See text for explanation.

probability for how likely adjacent nodes (e.g. pixels or atomic regions) belong to the same object or pattern. We then apply a transition kernel implementing split and merge dynamics (using the proposals). We refer to [2] for more detailed discussion.

For objects with little variability, such as the faces shown in Figure 20(b), we can use bottom-up proposals (e.g. AdaBoost) to activate a node that represents the entire face. The parsing graph can then be constructed downwards (i.e. in the decompositional mode) by expanding the face node to create child nodes for the parts of the face. These child nodes could, in turn, be expanded to grandchild nodes representing finer scale parts. The amount of node expansion can be made adaptive to depend on the resolution of the image. For example, the largest face in Figure 20(b) is expanded into child nodes but there is not sufficient resolution to expand the face nodes corresponding to the three smaller faces.

The major technical problem is to develop a mathematical criterion for which mode is most effective for which types of objects and patterns. This will enable the algorithm to adapt its search strategy accordingly.

2. Optimal ordering strategy for tests and kernels

The control strategy of our current image parsing algorithm does not select the tests and sub-kernels in an optimal way. At each time step, the choice of sub-kernel is independent of the current state W (though the choice of where in the graph to apply the sub-kernel will depend on W).

Moreover, bottom-up tests are performed which are never used by the algorithm.

It would be more efficient to have a control strategy which selects the sub-kernels and tests adaptively, provided the selection process requires low computational cost. We seek to find an optimal control strategy for selection which is effective for a large set of images and visual patterns. The selection criteria should select those tests and sub-kernels which maximize the gain in information.

We propose the two information criteria that we described in Section (2).

The first is stated in Theorem 1. It measures the information gained for variable w in the parsing graph by performing a new test Tst_+ . The information gain is $\delta(w|Tst_+) = KL(p(w|\mathbf{I}) || q(w|Tst(\mathbf{I}))) - KL(p(w|\mathbf{I}) || q(w|Tst_t(\mathbf{I}), F_+))$, where $Tst(\mathbf{I})$ denotes the previous tests (and KL is the Kullback-Leibler divergence).

The second is stated in Theorem 2. It measures the power of a sub-kernel \mathcal{K}_a by the decrease of the KL-divergence $\delta(\mathcal{K}_a) = KL(p || \mu_t) - KL(p || \mu_t \mathcal{K}_a)$. The amount of decrease δ_a gives a measure of the power of the sub-kernel \mathcal{K}_a when informed by $Tst_t(\mathbf{I})$.

We need also take into account the computational cost of the selection procedures. See [6] for a case study for how to optimally select tests taking into account their computational costs.

9 Summary and future work

This paper introduces a computational framework for parsing images into basic visual patterns. We formulated the problem using Bayesian probability theory and designed a stochastic DDMCMC algorithm to perform inference. Our framework gives a rigorous way to combine segmentation with object detection and recognition. We give proof of concept by implementing a model whose visual patterns include generic regions (texture and shading) and objects (text and faces). Our approach enables these different visual patterns to compete and cooperate to explain the input images.

This paper also provides a way to integrate discriminative and generative methods of inference. Both methods are extensively used by the vision and machine learning communities and correspond to the distinction between bottom-up and top-down processing. Discriminative methods are typically fast but can give sub-optimal and inconsistent results, see Figure 3. By contrast, generative methods are optimal (in the sense of Bayesian Decision Theory) but can be slow because

they require extensive search. Our DDMCMC algorithm integrates both methods, as illustrated in Figure 8, by using discriminative methods to propose generative solutions.

The goal of our algorithm is to construct a parse graph representing the image. The structure of the graph is not fixed and will depend on the input image. The algorithm proceeds by constructing Markov Chain dynamics, implemented by sub-kernels, for different moves to configure the parsing graph – such as creating or deleting nodes, or altering node attributes. Our approach can be scaled-up by adding new sub-kernels, corresponding to different vision models. This is similar in spirit to Ullman’s concept of “visual routines” [51]. Overall, the ideas in this paper can be applied to any other inference problem that can be formulated as probabilistic inference on graphs.

Other work by our group deals with a related series of visual inference tasks using a similar framework. This includes image segmentation [46], curve grouping [47], shape detection [48], motion analysis [2], and 3D scene reconstruction [23]. In the future, we plan to integrate these visual modules and develop a general purpose vision system.

Finally, we are working on ways to improve the speed of the image parsing algorithm as discussed in Section (8). In particular, we expect the use of the Swendsen-Wang cut algorithms [1, 2] to drastically accelerate the search. We anticipate that this, and other improvements, will reduce the running time of DDMCMC algorithms from 10-20 minutes [46] to well under a minute.

Appendix A: Proof of Theorem 2

[Proof]. For notational simplicity, we ignore the dependencies on the kernels and probabilities on the input image \mathbf{I} .

Let $\mu_t(W_t)$ be the state probability at time step t . After applying a sub-kernel $\mathcal{K}_a(W_{t+1}|W_t)$, its state becomes W_{t+1} with probability:

$$\mu_{t+1}(W_{t+1}) = \sum_{W_t} \mu_t(W_t) \mathcal{K}_a(W_{t+1}|W_t). \quad (17)$$

The joint probability $\mu(W_t, W_{t+1})$ for W_t and W_{t+1} can be expressed in two ways as:

$$\mu(W_t, W_{t+1}) = \mu_t(W_t) \mathcal{K}_a(W_{t+1}|W_t) = \mu_{t+1}(W_{t+1}) p_{\mathcal{MC}}(W_t|W_{t+1}), \quad (18)$$

where $p_{\mathcal{MC}}(W_t|W_{t+1})$ is the “posterior” probability for state W_t at time step t conditioned on state W_{t+1} at time step $t + 1$.

This joint probability $\mu(W_t, W_{t+1})$ can be compared to the joint probability $p(W_t, W_{t+1})$ at equilibrium (i.e. when $\mu_t(W_t) = P(W_t)$). We can also express $p(W_t, W_{t+1})$ in two ways:

$$p(W_t, W_{t+1}) = p(W_t)\mathcal{K}_a(W_{t+1}|W_t) = p(W_{t+1})\mathcal{K}_a(W_{t+1}, W_t), \quad (19)$$

where the second equality is obtained using the detailed balance condition on \mathcal{K}_a .

We calculate the Kullback-Leibler (K-L) divergence between the joint probabilities $P(W_t, W_{t+1})$ and $\mu(W_t, W_{t+1})$ in two ways using the first and second equalities in equations (18,19). We obtain two expressions (1 & 2) for the K-L divergence:

$$\begin{aligned} KL(p(W_t, W_{t+1}) || \mu(W_t, W_{t+1})) &= \sum_{W_{t+1}} \sum_{W_t} p(W_t, W_{t+1}) \log \frac{p(W_t, W_{t+1})}{\mu(W_t, W_{t+1})} \\ &\stackrel{1}{=} \sum_{W_t} p(W_t) \sum_{W_{t+1}} \mathcal{K}_a(W_{t+1}|W_t) \log \frac{p(W_t) \cdot \mathcal{K}_a(W_{t+1}|W_t)}{\mu_t(W_t)\mathcal{K}_a(W_{t+1}|W_t)} \\ &= KL(p(W) || \mu_t(W)) \end{aligned} \quad (20)$$

$$\begin{aligned} &\stackrel{2}{=} \sum_{W_{t+1}} \sum_{W_t} \mathcal{K}_a(W_t|W_{t+1}) p(W_{t+1}) \log \frac{p(W_{t+1})\mathcal{K}_a(W_t|W_{t+1})}{\mu_{t+1}(W_{t+1})p_{\mathcal{M}\mathcal{C}}(W_t|W_{t+1})} \\ &= KL(p(W_{t+1}) || \mu_{t+1}(W)) + E_{p(W_{t+1})}[KL(\mathcal{K}_a(W_t|W_{t+1}) || p_{\mathcal{M}\mathcal{C}}(W_t|W_{t+1}))] \end{aligned} \quad (21)$$

We equate the two alternative expressions for the KL divergence, using equations (20) and (21), and obtain

$$KL(p(W) || \mu_t(W)) - KL(p(W) || \mu_{t+1}(W)) = E_{p(W_{t+1})}[KL(\mathcal{K}_{a(t)}(W_t|W_{t+1}) || p_{\mathcal{M}\mathcal{C}}(W_t|W_{t+1}))]$$

This proves that the K-L divergence decreases monotonically. The decrease is zero only when $\mathcal{K}_{a(t)}(W_t|W_{t+1}) = p_{\mathcal{M}\mathcal{C}}(W_t|W_{t+1})$ (because $KL(p||\mu) \geq 0$ with equality only when $p = \mu$). This occurs if and only if $\mu_t(W) = p(W)$ (using the definition of $p_{\mathcal{M}\mathcal{C}}(W_t|W_{t+1})$, given in equation (18), and the detailed balance conditions for $\mathcal{K}_{a(t)}(W_t|W_{t+1})$). [End of Proof].

Appendix B: Classic Markov Chain convergence result

Traditional Markov chain convergence analysis is concerned with the total variance between the invariant (target) probability $p(W|\mathbf{I})$ and the Markov chain state probability $\mu_t(W)$,

$$\|\mu_t - p\|_{\text{TV}} = \sum_W |p(W|\mathbf{I}) - \mu_t(W)|. \quad (22)$$

In a finite state space Ω , a classic result relates the TV-measure to the second largest eigenvalue modulus ($\lambda_{\text{slem}} \in [0, 1)$) of the kernel \mathcal{K} . The following theorem is adopted from [15].

Theorem 4 (Diaconis and Hanlon, 1992) *For an irreducible kernel \mathcal{K} on finite space with invariant (target) probability $p(W|\mathbf{I})$ and initial state W_o , then at step n*

$$\|\mu_t - p\|_{\text{TV}} \leq \sqrt{\frac{1 - p(W_o|\mathbf{I})}{4p(W_o|\mathbf{I})}} \cdot \lambda_{\text{slem}}^t. \quad (23)$$

Appendix C: First-hitting-time analysis

We include a second result which bounds the expected first hitting-time $E[\tau(W)]$ for an arbitrary state W , where $\tau(W)$ is the number of time steps for a Markov Chain \mathcal{MC} to first visit a state $W \in \Omega$.

$$\tau(W) = \min\{t \geq 1 : W_t = W\}.$$

The expected first hitting time has higher resolution than the TV-norm in Theorem 4 or the KL-divergence in Theorem 2, as it is concerned with individual states rather than the entire space Ω . In particular, we are interested in the expected first hitting time for those states W with high probability $P(W)$. The following theorem [30] shows how an informed proposal q can improve the expected first hitting time for a special class of Markov Chains.

Theorem 5 (Maciucă and Zhu 2003) *Let $p(W), W \in \Omega$ be the invariance (target) probability of a Markov chain \mathcal{MC} , and $Q(W, W') = q(W')$ be the proposal probability in the Metropolis-Hasting equation (8), then*

$$\frac{1}{\min\{p(W), q(W)\}} \leq E[\tau(W)] \leq \frac{1}{\min\{p(W), q(W)\}} \cdot \frac{1}{1 - \|p - q\|_{\text{TV}}} \quad (24)$$

where $\|p - q\| = \frac{1}{2} \sum_{x \in \Omega} |p(x) - q(x)| \leq 1$.

In this Markov Chain design, the proposal probability depends only on W' not W , and is called the Metropolized independence sampler (MIS). A similar result can also be obtained for the Metropolized Gibbs sampler (MGS). This result shows that we should choose the proposal probability q so that it overlaps with p and so that $q(W)$ is large for those states W with high probability $P(W)$. By contrast, choosing q to be the uniform probability, $q(W) = 1/|\Omega| \forall W \in \Omega$, will result in expected first hitting times of length greater than $|\Omega|$ for any state with $P(W) > 1/|\Omega|$.

Appendix D: Multiple routes in the sub-kernels

This section addresses a practical issue about detailed balance which leads to a trade-off between computation and efficiency in the algorithm design. For notational simplicity, we write terms like $Q(W'|W : \mathbf{I})$ & $\mathcal{K}(W'|W : \mathbf{I})$ as $Q(W, W')$ & $\mathcal{K}(W, W')$ in this section.

In some situations, the sub-kernels can give multiple routes between states W and W' . For example, one region can be split into the same two sub-regions following different bottom-up proposals. In this section, we show that detailed balance can be maintained either by using extra computation to integrate out the multiple routes, or by treating the alternative routes separately using different sub-kernels but at the price of reduced efficiency.

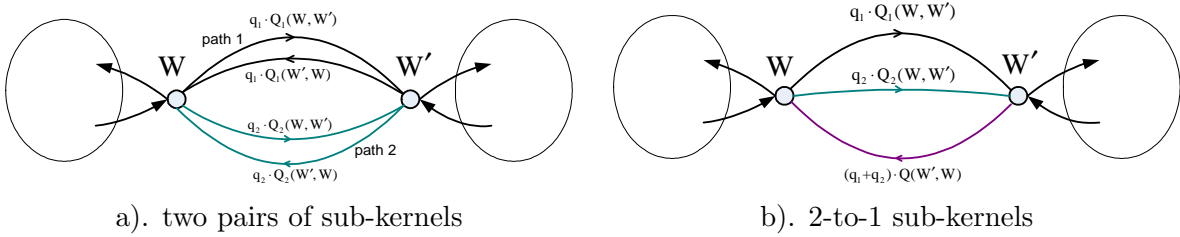


Figure 21: There may be multiple routes between the two states W and W' and this creates an additional computational burden for computing the acceptance rates.

Suppose there are n pairs of routes $\{(q(i)Q_i(W, W'), q(i)Q_i(W', W), i = 1..n\}$ between two states W and W' , such as the two pairs of moves in Figure 21a. We represent the transition kernel by

$$\mathcal{K}(W, W') = Q(W, W')\alpha(W, W'), \quad (25)$$

where

$$Q(W, W') = \sum_{i=1}^n q(i)Q_i(W, W'),$$

and

$$\alpha(W, W') = \min\left(1, \frac{Q(W', W)p(W')}{Q(W, W')p(W)}\right).$$

Every time a move from W to W' is considered we have to consider all possible routes, which can result in heavy computation.

Instead, we can treat each route as a sub-kernel as described in section (2.3). Then the Markov chain satisfies detailed balance because each pair $\mathcal{K}_i(W, W')$ and $\mathcal{K}_i(W', W)$ are speci-

fied by Metropolis-Hastings 8 and hence obey the detailed balance equation $p(W)\mathcal{K}_i(W, W') = p(W')\mathcal{K}_i(W', W), i = 1..n$. This reduces the computational cost. But, as the following theorem shows, this reduction comes at the price of efficiency.

Theorem 6 *The kernels $\mathcal{K}(W, W')$ and $\mathcal{K}(W', W)$ as computed in eqn. (25) satisfy*

$$\mathcal{K}(W, W') \geq \sum_{i=1}^n q(i)\mathcal{K}_i(W, W'), \text{ and } \mathcal{K}(W', W) \geq \sum_{i=1}^n q(i)\mathcal{K}_i(W', W), \forall W' \neq W.$$

This theorem says that the Markov chain which obeys detailed balance for each pair of moves is less effective than the one which combines all the routes. Markov chain design must balance the computation cost of computing all the proposals against the effectiveness of the Markov kernel. The situation shown in Figure 21b can be treated as a special case of Figure (21a).

Acknowledgments

This work is supported by an NIH (NEI) grant RO1-EY 012691-04, an NSF SGER grant IIS-0240148, an NSF grant IIS-0244763 and an ONR grant N00014-02-1-0952. The authors thank the Smith-Kettlewell research institute for providing us with text training images. We thank Yingnian Wu for stimulating discussions on the Markov chain convergence theorems.

References

- [1] A. Barbu and S.C. Zhu, “Graph partition by Swendsen-Wang cut”, *Proc. of Int’l Conf. on Computer Vision*, Nice, France, October, 2003.
- [2] A. Barbu and S.C. Zhu, “Multi-grid and multi-level Swendsen-Wang cuts for hierarchic graph partition”, *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, Washington DC, June, 2004.
- [3] K. Barnard and D.A. Forsyth, “Learning the semantics of words and pictures”, *ICCV*, 2001.
- [4] S. Belongie, J. Malik, and J. Puzicha. “Shape matching and object recognition using shape contexts”, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24:509–522, 2002.
- [5] E. Bienenstock, S. Geman, and D. Potter, “Compositionality, MDL Priors, and Object Recognition”, NIPS, 1997.
- [6] G. Blanchard and D. Geman, “Hierarchical testing designs for pattern recognition,” Technical report, Math. Science, Johns Hopkins University, 2003.
- [7] P. Bremaud, “Markov Chains: Gibbs Fields, Monte Carlo Simulation and Queues,” Springer, 1999. (Chapter 6).
- [8] K.W. Bowyer, C. Kranenburg, and S. Dougherty, “Edge detector evaluation using empirical ROC curves”, *Computer Vision and Image Understanding*, 84, no. 1, pp.77-103, Oct. 2001.

- [9] J. Canny, “A computational approach to edge detection”, *IEEE Trans. on PAMI*, vol.8, no.6, nov. 1986.
- [10] X. Chen and A.L. Yuille. “AdaBoost Learning for Detecting and Reading Text in City Scenes”. *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*. Washington DC, June, 2004.
- [11] T. F. Cootes, G. J. Edwards, and C. J. Taylor, “Active appearance models”, *IEEE Trans. on PAMI* no. 23, vol. 6, 2001.
- [12] D. Comaniciu and P. Meer, “Mean Shift Analysis and Applications,” *Proc. of ICCV*, 1999.
- [13] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, (pp 33-36) John Wiley and Sons, Inc, NY, 1991.
- [14] P. Dayan, G. Hinton, R. Neal and R. Zemel. “The Helmholtz Machine”. *Neural Computation*, 7, pp 889-904. 1995.
- [15] P. Diaconis and P. Hanlon, “Eigenanalysis for some examples of the Metropolis algorithms”, *Contemporary Mathematics*, vol. 138, pp 99-117, 1992.
- [16] H. Drucker, R. Schapire, and P. Simard, “Boosting performance in neural networks,” *Intl J. Pattern Rec. and Artificial Intelligence*, vol. 7, no. 4, 1993.
- [17] C. Fowlkes and J. Malik, “How Much Does Globalization Help Segmentation?”, *CVPR 2004*.
- [18] Y. Freund and R. Schapire, “Experiments with a new boosting algorithm”, *Proc. of 13th Int’l Conference on Machine Learning*, 1996.
- [19] J. Friedman, T. Hastie and R. Tibshirani, “Additive logistic regression: a statistical view of boosting”, Dept. of Statistics, Stanford Univ. Technical Report. 1998.
- [20] S. Geman and C.R. Huang, “Diffusion for global optimization”, *SIAM J. on Control and Optimization*, vol. 24, no. 5, 1986.
- [21] P. J. Green, “Reversible Jump Markov Chain Monte Carlo Computation and Bayesian Model Determination,” *Biometrika*, vol. 82, no. 4, pp. 711-732, 1995.
- [22] P. Hallinan, G. Gordon, A. Yuille, P. Giblin, and D. Mumford, *Two and Three Dimensional Patterns of the Face*, A.K. Peters, 1999.
- [23] F. Han and S.C. Zhu, “Bayesian reconstruction of 3D shapes and scenes from a single image”, *Proc. Int’l Workshop on High Level Knowledge in 3D Modeling and Motion*, Nice France, October, 2003.
- [24] W.K. Hastings, “Monte Carlo sampling methods using Markov chains and their applications”, *Biometrika*, 57, 97-109, 1970.
- [25] D. Klein and C. D. Manning, “A generative constituent-context model for improved grammar induction”, *Proc. of 40th Annual Meeting of the Assoc. for Computational Linguistics*, July 2002.
- [26] S. Konishi, J. M. Coughlan, A. L. Yuille, and S. C. Zhu, “Statistical edge detection: learning and evaluating edge cues”, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 25, no.1, pp 57-74, 2003.
- [27] F.F. Li R. VanRullen, C. Koch, and P. Perona, “Rapid natural scene categorization in the near absence of attention”, *Proc. of National Academy of Sciences*, vol.99, no.14, 2003.
- [28] J.S. Liu, “Monte Carlo Strategies in Scientific Computing”, Springer, 2001
- [29] L.D. Lowe, “Distinctive image features from scale-invariant keypoints”, *IJCV*, 2003.

- [30] R. Maciucă and S.C. Zhu, “How Do Heuristics Expedite Markov Chain Search,” *Proc. of 3rd Workshop on Statistical and Computational Theory for Vision*, 2003.
- [31] J. Malik, S. Belongie, T. Leung and J. Shi, “Contour and texture analysis for image segmentation”, *Int’l Journal of Computer Vision*, vol.43, no.1, 2001.
- [32] C.D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press. 2003.
- [33] D. Marr. *Vision*. W.H. Freeman and Co. San Francisco, 1982.
- [34] D. Martin, C. Fowlkes, D. Tal and J. Malik, “A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics”, *Proc. of 8th Int’l Conference on Computer Vision*, 2001.
- [35] N. Metropolis, M.N. Rosenbluth, A.W. Rosenbluth, A.H. Teller, and E. Teller, “Equations of State Calculations by Fast Computing Machines”, *J. Chem. Phys.* 21, 1087-92, 1953.
- [36] B. Moghaddam and A. Pentland, “Probabilistic visual learning for object representation”, *IEEE Trans. PAMI*, vol.19, no.7, 1997.
- [37] D.B. Mumford. “Neuronal Architectures for Pattern-theoretic Problems”. In *Large-Scale Neuronal Theories of the Brain*. Eds. C. Koch and J. L. Davis. MIT Press. A Bradford Book. 1995.
- [38] K. Murphy, A. Torralba, and W.T. Freeman, “Using the forest to see the tree: a graphical model relating features, objects and the scenes”, *NIPS*, 2003.
- [39] W. Niblack. *An Introduction to Digital Image Processing*. pp. 115-116, Prentice Hall, 1986.
- [40] P. J. Phillips, H. Wechsler, J. Huang, and P. Rauss, “The FERET database and evaluation procedure for face recognition algorithms”, *Image and Vision Computing Journal*, vol. 16, no. 5, 1998.
- [41] J. Ponce, S. Lazebnik, F. Rothganger, and C. Schmid, “Toward true 3D object recognition”, *Reconnaissance de Formes et Intelligence Artificielle*, Toulouse, FR. 2004.
- [42] R. E. Schapire, “The boosting approach to machine learning: an overview”, *MSRI Workshop on Non-linear Estimation and Classification*, 2002.
- [43] J. Shi and J. Malik, “Normalized Cuts and Image Segmentation,” *IEEE Trans. PAMI*, vol. 22, no. 8, Aug. 2000.
- [44] S. Thorpe, D. Fize, and C. Marlot, “Speed of processing in the human visual system”, *Nature*, vol. 381 (6582), 520-522, Janue, 1996.
- [45] A. Treisman, “Features and objects in visual processing”, *Scientific American*, November, 1986.
- [46] Z. Tu and S.C. Zhu, “Image segmentation by Data-driven Markov chain Monte Carlo”, *IEEE Trans. PAMI*, vol. 24, no.5, pp. 657-673, 2002.
- [47] Z.W. Tu and S.C. Zhu, “Parsing images into regions, curves and curve groups”, *Int’l Journal of Computer Vision*, (Under review), A short version appeared in the *Proc. of ECCV*, 2002.
- [48] Z.W. Tu and A.L. Yuille. “Shape Matching and Recognition: Using Generative Models and Informative Features”. In *Proceedings European Conference on Computer Vision. ECCV’04*. Prague. 2004.
- [49] M. Turk and A. Pentland, “Eigenfaces for recognition,” *J. of Cognitive Neurosciences*, vol.3, no.1, pp. 71-86, 1991.
- [50] S. Ullman, “Visual routines”, *Cognition*, vol.18, pp.97-159, 1984.

- [51] S. Ullman, "Sequence Seeking and Counterstreams: A Model for Bidirectional Information Flow in the Cortex". In *Large-Scale Neuronal Theories of the Brain*. Eds. C. Koch and J. L. Davis. MIT Press. A Bradford Book. 1995.
- [52] P. Viola and M. Jones, "Fast and robust classification using asymmetric Adaboost and a detector cascade", In *Proc. of NIPS01*, 2001.
- [53] M. Weber, M. Welling and P. Perona, "Towards Automatic Discovery of Object Categories", *Proc. of CVPR*, 2000.
- [54] J. Wu, J.M. Rehg, and M.D. Mullin, "Learning a rare event detection cascade by direct feature selection", NIPS, 2004.
- [55] J.S. Yedidia, W.T. Freeman, and Y. Weiss, "Generalized belief propagation". In *Advances in Neural Information Processing Systems 13*, pp 689-695. 2001.
- [56] S. C. Zhu and A. L. Yuille, "Region competition: unifying snakes, region growing, and Bayes/MDL for multiband image segmentation," *IEEE Trans. PAMI*, vol. 18, no. 9, 1996.
- [57] S.C. Zhu, R. Zhang, and Z.W. Tu, "Integrating top-down/bottom-up for object recognition by data-driven Markov chain Monte Carlo", *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, Hilton Head, SC. 2000.